PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-121490
(43)Date of publication of application : 12.05.1995

(51)Int.Cl. G06F 15/16

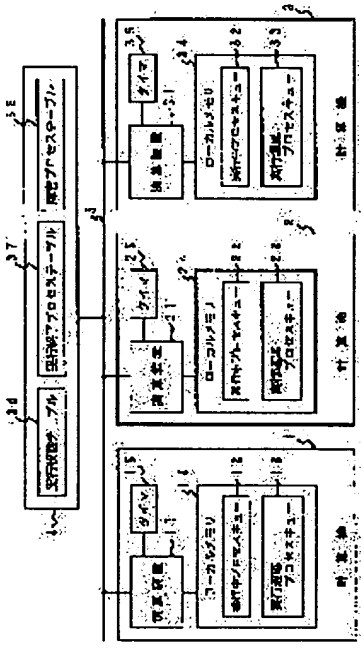
(21)Application number : 06-085544
(22)Date of filing : 31.03.1994

(71)Applicant : TOSHIBA CORP
(72)Inventor : HASEGAWA TETSUO
OIYAKE YASUKUNI

(30)Priority
Priority number : 05216282 Priority date : 31.08.1993 Priority country : JP

(54) MULTIPLE PROCESSING SYSTEM AND PROGRAM EXECUTION CONTROL METHOD

(57)Abstract:
PURPOSE: To prevent the down of a whole computer because of a program bug by removing application process executed in a precedent system computer becomes down from the application process group of the ensuing system computers and permits the ensuing system computer to be operated as the advance system ones.
CONSTITUTION: The special application process group is execution-started by the ensuing system computers 2 and 3 at the point of time when it is delayed for a period fulfilling a prescribed condition from the point of time the execution of the precedent system computer 1 is started. Then, when the precedent system computer 1 becomes down, application process which is executed in the precedent system computer 1 is evaded from the application process group of the ensuing system computers 2 and 3 so as to permit the ensuing system computers 2 and 3 to be operated as the precedent system ones. Thus, the down of the whole computers 1-3 because of the program bug is prevented and the processing of a whole system can be continued.



LEGAL STATUS

[Date of request for examination] 11.09.2000
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number] 3483931

[Date of registration] 17.10.2003

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報(A)　(11)特許出願公開番号

特開平7－121490

(43)公開日　平成7年(1995)5月12日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　15/16	４７０　Ｂ	7429－5Ｌ		

審査請求　未請求　請求項の数8　ＦＤ　(全 32 頁)

(21)出願番号　特願平6－85544

(22)出願日　平成6年(1994)3月31日

(31)優先権主張番号　特願平5－216282

(32)優先日　平5(1993)8月31日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72)発明者　長谷川　哲夫
神奈川県川崎市幸区柳町70番地　株式会社東芝柳町工場内

(72)発明者　岡宅　泰邦
神奈川県川崎市幸区柳町70番地　株式会社東芝柳町工場内

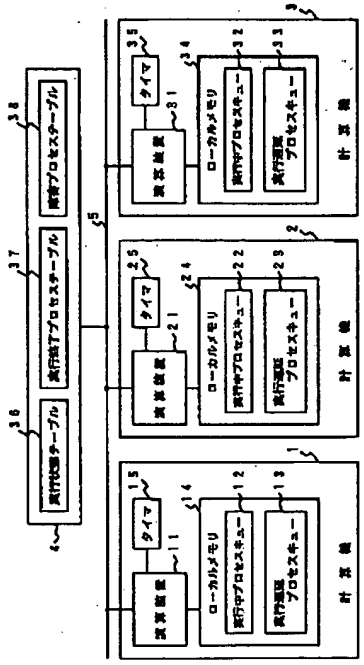
(74)代理人　弁理士　鈴江　武彦

(54)【発明の名称】　多重処理システムおよびプログラム実行制御方法

(57)【要約】

【目的】プログラムバグで計算機に障害が発生した場合でも、全計算機がダウンするのを避けることのできる多重処理システムを提供する。

【構成】先行系と追従系とに分けられた複数の計算機１，２，３と、先行系の計算機１で特定のアプリケーションプロセス群を実行開始させるとともに先行系の計算機１の実行開始時点より所定の条件を満たす期間だけ遅れた時点から追従系の計算機２，３で上記特定のアプリケーションプロセス群を実行開始させる手段と、先行系の計算機１がダウンしたときには、ダウンしたときに先行系の計算機１で実行していたアプリケーションプロセスを追従系の計算機２，３のアプリケーションプロセス群の中から取り除いて追従系の計算機２，３を先行系として動作させる手段とを有する多重処理システム。



1

【特許請求の範囲】

【請求項1】先行系と追従系とに分けられた複数の計算機と、

前記先行系の計算機で特定のアプリケーションプロセス群を実行開始させるとともに、前記先行系の計算機の実行開始時点より所定の条件を満たす期間だけ遅れた時点から前記追従系の計算機で前記特定のアプリケーションプロセス群を実行開始させる手段と、

前記先行系の計算機がダウンしたときには、ダウンしたときに前記先行系の計算機で実行していたアプリケーションプロセスを前記追従系の計算機のアプリケーションプロセス群の中から取り除いて前記追従系の計算機を先行系として動作させる手段とを具備してなることを特徴とする多重処理システム。

【請求項2】先行系と追従系とに分けられた複数の計算機と、

前記先行系の計算機で特定のアプリケーションプロセス群を実行開始させるとともに、前記先行系の計算機の実行開始時点より所定の条件を満たす期間だけ遅れた時点から前記追従系の計算機で前記特定のアプリケーションプロセス群を実行開始させる手段と、

前記先行系の計算機がダウンしたときには、ダウンしたときに前記先行系の計算機で実行していたアプリケーションプロセスを前記追従系の計算機のアプリケーションプロセス群の中から取り除くとともに前記追従系を構成している計算機を新先行系と新追従系とに再構成し、前記新先行系の計算機側から残りのアプリケーションプロセスを実行開始させる手段とを具備してなることを特徴とする多重処理システム。

【請求項3】同一の機能を有する複数の版のアプリケーションプロセス用プログラムを保持する先行系と追従系とに分けられた複数の計算機と、

前記先行系の計算機で特定のアプリケーションプロセス群をそれぞれ特定の版のプログラムに従って実行開始させるとともに、前記先行系の計算機の実行開始点より所定の条件を満たす期間だけ遅れた時点から前記追従系の計算機で前記特定のアプリケーションプロセス群をそれぞれ前記特定の版のアプリケーションプロセス用プログラムに従って実行開始させる手段と、

前記先行系の計算機がダウンしたときには、ダウンしたときに前記先行系の計算機で実行していたアプリケーションプロセスを前記特定の版とは異なる版のプログラムに従って前記追従系の計算機で実行させる手段とを具備してなることを特徴とする多重処理システム。

【請求項4】同一の機能を有する複数の版のアプリケーションプロセス用プログラムを保持する先行系と追従系とに分けられた複数の計算機と、

前記先行系の計算機で特定のアプリケーションプロセス群をそれぞれ特定の版のプログラムに従って実行開始させるとともに、前記先行系の計算機の実行開始点より所定の条件を満たす期間だけ遅れた時点から前記追従系の計算機で前記特定のアプリケーションプロセス群をそれぞれ前記特定の版のアプリケーションプロセス用プログラムに従って実行開始させる手段と、

前記先行系の計算機がダウンしたときには、ダウンしたときに前記先行系の計算機で実行していたアプリケーションプロセスを前記特定の版とは異なる版のプログラムに従って前記追従系の計算機で実行させるとともに前記追従系を構成している計算機を新先行系と新追従系とに再構成し、前記新先行系の計算機側から残りのアプリケーションプロセスを実行開始させる手段とを具備してなることを特徴とする多重処理システム。

【請求項5】第1のプログラムとこれら第1のプログラムの実行を制御するオペレーティングシステムを搭載した少なくとも一つの計算機におけるプログラムの実行を制御するプログラム実行制御方法において、

前記第1のプログラムを少なくとも該第1のプログラムのプログラム名と該第1のプログラムより低能力でかつバグの潜在確率がより低い第2のプログラムのプログラム名リスト情報を伴って起動または呼出し、前記第1のプログラムが停止したときは前記プログラム名リスト情報に従って前記第2のプログラムを起動または呼出すことを特徴とするプログラム実行制御方法。

【請求項6】第1のプログラムとこれら第1のプログラムの実行を制御するオペレーティングシステムをそれぞれ搭載した複数の計算機から構成される計算機システムにおけるプログラムの実行を制御するプログラム実行制御方法において、

前記複数の計算機のうちの一つの計算機に搭載された第1のプログラムを少なくとも該第1のプログラムのプログラム名と該第1のプログラムより低能力でかつバグの潜在確率がより低い第2のプログラムのプログラム名リスト情報を伴って起動または呼出すとともに、

該第1のプログラムのプログラム名が前記一つの計算機内のオペレーティングシステムにより該計算機内に登録されていないと確認されたときは、該オペレーティングシステムが他の計算機に対して該第1のプログラムが存在するか問い合わせ、

前記他の計算機に該第1のプログラムが存在しているときは、該他の計算機内のオペレーティングシステムに対して該第1のプログラムの起動ないし呼出し制御を依頼し、

前記他の計算機内にも該第1のプログラムが存在しないときは、前記プログラム名リスト情報に従って前記第2のプログラムを起動または呼出すことを特徴とするプログラム実行制御方法。

【請求項7】第1のプログラムとこれら第1のプログラムの実行を制御するオペレーティングシステムをそれぞれ搭載した複数の計算機から構成される計算機システムにおけるプログラムの実行を制御するプログラム実行制

3

御方法において、
前記複数の計算機のうちの一つの計算機に搭載された第１のプログラムを、少なくとも該第１のプログラムのプログラム名と該第１のプログラムより低能力でかつバグの潜在確率がより低い第２のプログラムのプログラム名リスト情報を伴って起動または呼出し、
前記複数の計算機に登録された前記第１のプログラムの中で実行が異常終了した場合に、該第１のプログラムのプログラム名を前記複数の計算機内の全てのオペレーティングシステムに登録するとともに、
該オペレーティングシステムは、前記プログラム名リスト情報に従って前記第２のプログラムを起動または呼出すことを特徴とするプログラム実行制御方法。
【請求項８】前記第２のプログラムのプログラム名リスト情報は、実行優先度情報が付加されていることを特徴とする請求項５乃至７のいずれかに記載のプログラム実行制御方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は、複数の計算機で同一の処理ないし同一の機能を有する処理を並行して実行する多重処理システムおよびプログラム実行制御方法に係り、特にプログラムのバグによって全計算機がダウンするのを防止できるようにした多重処理システムおよびプログラム実行制御方法に関する。
【０００２】
【従来の技術】変動するデータを処理し、その処理結果を化学・鉄鋼プラントのような産業システム、交通制御システム、あるいは原子力プラントのような電力システムといった制御対象システムに伝達制御するシステムにおいては、いかなる状況下にあってもシステムを常に安全に制御し、システムに与えられたミッションを確実に達成することが要求される。このような要求に対し、複数の計算機により同一処理または同一機能を有する処理を並行して実行する多重処理システムが従来から利用されている。多重処理システムには、以下の示す種々の方式がある。
【０００３】（１）複数の計算機で同一のアプリケーションプロセス群を実行する方式。
【０００４】複数の計算機に同一のアプリケーションプロセス群を実行させる多重処理システムでは、たとえ１台の計算機が何らかの障害によってダウンしても、他の計算機で処理を続行させることができるので、処理の中断を避けることができる。この方式の多重処理システムは、いずれか１台の計算機にハードウェア障害が発生する確率に比べて複数の計算機に同時にハードウェア障害が発生する確率が非常に低いことを有効に利用している。
【０００５】しかしながら、アプリケーションプロセスのプログラムが完全であるという保証はない。プログラ

4

ムバグを含むアプリケーションプロセスを実行すると、計算機に障害が発生し、この障害は全ての計算機で起こる。従って、たとえ並列多重処理を行っていても、システムでの処理が中断することになる。
【０００６】たとえば、図１５（ａ）に示すように、各タイムスライス毎にアプリケーションプロセスＰ１～Ｐ４が投入された場合、プログラムバグのないときには、この図に示すようなタイミングでアプリケーションプロセスが生成され実行される。なお、タイムスライス回数とは何回目のタイムスライスが起きたかを示し、これはすなわち時刻の経過を意味する。また、この例において、Ｐ２はＰ１により、Ｐ３，Ｐ４はＰ２によってそれぞれ生成される。ここで、仮にＰ３のプロセスのバグによって計算機に障害が起こり、計算機がダウンすると、従来の並列多重処理システムあるいは１台の計算機で実行した場合には、図１５（ｂ）に示すタイムスライス回数４の時点で全計算機がダウンする。従って、プロセスＰ４が正常に動くことの可能な場合であっても、このプロセスＰ４の処理を実行できないことになる。
【０００７】小型の計算機や制御用に用いられる高速応答性を重視した計算機においては、計算機の動きを管理するＯＳ（オペレーティングシステム）等の保護機構が弱く、アプリケーションプロセスのプログラムバグにより障害の発生する可能性が高い。従って、並列多重処理を行っても、一部のアプリケーションプロセスのプログラムバグでシステム全体の処理が中断してしまうという問題があった。
【０００８】（２）複数の計算機で同一の機能を有する複数の版のプログラムに従ってアプリケーションプロセスを並行に実行する方式。
【０００９】複数の計算機に同一の機能を有する複数の版（バージョン）のプログラム構造にもとづくアプリケーションプロセスを並行に実行させる多重処理システムでは、たとえ１台の計算機が何らかの障害によってダウンするか、または内部状態に矛盾を起こしたとき外部に悪影響を及ぼすことを避けるために以降の処理を中断しても、他の計算機で処理を続行させることができるので、処理の中断を避けることができる。
【００１０】この方式の多重処理システムは、いずれか１台の計算機にハードウェア障害が発生する確率が複数台の計算機に同時に発生する確率に比べて非常に少ないことに加えて、同一機能を有するアプリケーションプロセスを有するいずれかの版のプログラムにプログラムバグが存在して障害が発生しても、同一の機能を有する他の版のプログラムにはプログラムバグが存在せず、障害が発生しない可能性が高い点を有効に利用している。
【００１１】この方式の並列処理システムにおいて、システム全体の処理が中断されないためには、同一の機能を有する複数の版のプログラムのうち最低１つの版は障害が発生しないことが条件であるが、障害が発生しにく

5

い安全な作り方をしたいわゆる安全版プログラムは実行時間が余分にかかる場合が非常に多く、その結果、このような安全版プログラムに従って処理を実行させると、システム全体としての処理時間が遅くなってしまうという問題がある。

【0012】安全版プログラムの実行時間が余分にかかる理由は、主に以下のようなものがある。第1にプログラム中の随所に障害発生を防ぐための異常検出処理を追加しなくてはならないこと、第2に時間がかかる処理の代表である検索処理において不要な検索を防ぐための処理を追加することによって処理時間を短縮する手法が各種考案されているように、一般に高速化するためには余分な処理が必要になり、その分プログラムバグが発生し易くなること、などである。

【0013】（3）複数の計算機内で稼働するプログラム全体をN版プログラム方式によって多重化する方式。

【0014】プログラムの多重化は、本来プログラムを複製して複数の計算機で実行する方式であるから、プログラムにバグが内在していれば、多重化していても共通のバグが原因で計算の停止やシステムの一部機能の停止を引き起こす原因となりシステムに与えられたミッションを達成することが不可能になる。

【0015】このような事態を回避するための一つの方式として、"N版プログラム方式"がある。この方式は（2）の方式と類似しているが、特に同一機能を達成する複数の版（バージョン）のプログラムを別々の設計者が異なる手順で互いに隔離された環境下で作成することが特徴である。そして、このようにして作成された同一機能を果たす異なったプログラム群を複数の計算機内で並列に実行し、それにより得られた複数の出力結果のうち過半数が一致したものを正しい出力結果として選択する。

【0016】この"N版プログラム方式"は、複数板のプログラムモジュール群を相互に隔離された複数の設計者で作成するため、プログラムモジュール数が多くなるほどプログラム作成コストの非常な増大とともに保守管理のコストの非常な増大を引き起こすことになる。例えば、3チームが独立に異なる手順で同一機能プログラムを開発する場合、従来と同一水準の品質を保証するプログラムを開発するには3倍の開発人員が必要となり、保守の観点からも3つの版のプログラム保守・管理のコスト増大は避けられない。また、3つの版のプログラムを並列に動作させて結果を判定する場合、最も処理の遅いプログラムの処理性能でシステム性能が決まってしまうことになる。

【0017】

【発明が解決しようとする課題】上述したように、従来の多重処理システムはハードウェア障害に対しては対処できるが、プログラムのバグに対する有効性は十分でなかった。

6

【0018】本発明の目的は、ハードウェア障害はもとより、プログラムのバグで計算機に障害が発生した場合でも、全計算機がダウンするのを避けてシステム全体の処理を続行できる多重処理システムおよびプログラム実行制御方法を提供することにある。

【0019】

【課題を解決するための手段】上記目的を達成するために、第1の発明に係る多重処理システムは、先行系と追従系とに分けられた複数の計算機と、前記先行系の計算機で特定のアプリケーションプロセス群を実行開始させるとともに前記先行系の計算機の実行開始時点より所定の条件を満たす期間だけ遅れた時点から前記追従系の計算機で前記特定のアプリケーションプロセス群を実行開始させる手段と、前記先行系の計算機がダウンしたときには、ダウンしたときに前記先行系の計算機で実行していたアプリケーションプロセスを前記追従系の計算機のアプリケーションプロセス群の中から取り除いて前記追従系の計算機を先行系として動作させる手段とを備えることを特徴とする。

【0020】また、このような基本構成において先行系の計算機がダウンしたとき、ダウンした前記先行系の計算機で実行していたアプリケーションプロセスを追従系の計算機のアプリケーションプロセス群の中から取り除くとともに追従系を構成している計算機を新先行系と新追従系とに再構成し、新先行系の計算機側から残りのアプリケーションプロセスを実行開始させる手段を設けることを特徴とする。

【0021】第2の発明に係る多重処理システムは、同一の機能を有する複数の版のアプリケーションプロセス用プログラムを保持する先行系と追従系とに分けられた複数の計算機と、前記先行系の計算機で特定のアプリケーションプロセス群をそれぞれ特定の版のプログラムに従って実行開始させるとともに、前記先行系の計算機の実行開始点より所定の条件を満たす期間だけ遅れた時点から前記追従系の計算機で前記特定のアプリケーションプロセス群をそれぞれ前記特定の版のアプリケーションプロセス用プログラムに従って実行開始させる手段と、前記先行系の計算機がダウンしたときには、ダウンしたときに前記先行系の計算機で実行していたアプリケーションプロセスを前記特定の版とは異なる版のプログラムに従って前記追従系の計算機で実行させる手段とを備えたことを基本的な特徴とする。

【0022】また、このような基本構成において、先行系の計算機がダウンしたときには、ダウンしたときに前記先行系の計算機で実行していたアプリケーションプロセスを前記特定の版とは異なる版のプログラムに従って前記追従系の計算機で実行させるとともに前記追従系を構成している計算機を新先行系と新追従系とに再構成し、前記新先行系の計算機側から残りのアプリケーションプロセスを実行開始させることを特徴とする。

7

【0023】第3の発明に係るプログラム実行制御方法は、第1のプログラムとこれら第1のプログラムの実行を制御するオペレーティングシステムを搭載した少なくとも一つの計算機におけるプログラムの実行を制御するプログラム実行制御方法において、前記第1のプログラムを少なくとも該プログラムのプログラム名と該プログラムより低能力でかつバグの潜在確率がより低い第2のプログラムのプログラム名リスト情報を伴って起動または呼出し、前記第1のプログラムが停止したときは前記プログラム名リスト情報に従って前記第2のプログラムを起動または呼出すことを特徴とする。

【0024】また、このような基本構成において、第1のプログラムとこれら第1のプログラムの実行を制御するオペレーティングシステムをそれぞれ搭載した複数の計算機から構成される計算機システムにおけるプログラムの実行を制御する場合は、前記複数の計算機のうちの一つの計算機に搭載された第1のプログラムを少なくとも該第1のプログラムのプログラム名と該第1のプログラムより低能力でかつバグの潜在確率がより低い第2のプログラムのプログラム名リスト情報を伴って起動または呼出すとともに、該第1のプログラムのプログラム名が前記一つの計算機内のオペレーティングシステムにより該計算機内に登録されていないと確認されたときは、該オペレーティングシステムが他の計算機に対して該第1のプログラムが存在するか問い合わせ、前記他の計算機に該第1のプログラムが存在しているときは、該他の計算機内のオペレーティングシステムに対して該第1のプログラムの起動ないし呼出し制御を依頼し、前記他の計算機内にも該第1のプログラムが存在しないときは、前記プログラム名リスト情報に従って前記第2のプログラムを起動または呼出すことを特徴とする。

【0025】さらに、前記複数の計算機のうちの一つの計算機に搭載された第1のプログラムを、少なくとも該第1のプログラムのプログラム名と該第1のプログラムより低能力でかつバグの潜在確率がより低い第2のプログラムのプログラム名リスト情報を伴って起動または呼出し、前記複数の計算機に登録された前記第1のプログラムの中で実行が異常終了した場合に、該第1のプログラムのプログラム名を前記複数の計算機内の全てのオペレーティングシステムに登録するとともに、該オペレーティングシステムは、前記プログラム名リスト情報に従って前記第2のプログラムを起動または呼出すことを特徴とする。

【0026】また、前記第2のプログラムのプログラム名リスト情報に実行優先度情報を付加しておき、この実行優先度情報に従って第2のプログラムを起動または呼出すようにすることも有効である。

【0027】

【作用】アプリケーションプロセスのプログラムバグによって全ての計算機がダウンするのを防止するには、少

8

なくとも1台の計算機がそのアプリケーションプロセスの実行を避ける必要がある。

【0028】そこで、第1の発明に係る多重処理システムでは、計算機群の一部を先行系、残りを追従系とに分け、追従系の計算機群では発生したアプリケーションプロセスを保存しておき、先行系の計算機から指定した条件（例えば実行終了）を満たした旨の通知があるまで実行を延期することにより、そのアプリケーションプロセスが計算機をダウンさせるプログラムバグを含まないことを確認してから実行させる。一方、先行系の計算機がダウンしたことを検出した場合には、追従系の計算機側では保存してあるアプリケーションプロセスからダウンの原因となったアプリケーションプロセスを取り除き、追従系の計算機がダウンするのを防ぐ。これにより、アプリケーションプロセスのプログラムバグによって全計算機がダウンするのを防止でき、処理を続行させることが可能となる。

【0029】また、先行系の計算機がダウンした場合に、追従系を構成している計算機を新先行系と新追従系とに再構成すると、常に先行系と追従系の両系の計算機を存在させることができ、次に先行系の計算機がダウンした場合に対処することが可能となる。

【0030】一方、アプリケーションプロセス群のそれぞれに対し、処理時間は短いが障害を発生するプログラムバグを含む可能性の高い高速版プログラムと、処理時間は長いが障害を発生するプログラムバグを含む可能性の少ない安全版プログラムとを用意した場合、高速版プログラムにプログラムバグが含まれないことが明らかなアプリケーションプロセスは、全ての計算機で高速版のプログラムに従って実行可能である。しかし、高速版プログラムにプログラムバグが含まれる可能性の大きいアプリケーションプロセスは、少なくとも1台の計算機で安全版プログラムに従ってアプリケーションプロセスを実行し、全計算機がダウンすることを避ける必要がある。

【0031】そこで、第2の本発明に係る多重処理システムでは、それぞれの計算機がアプリケーションプロセス群に対し同一機能を持つ高速版プログラムと安全版プログラムを持ち、計算機群の一部を先行系、残りを追従系とに分け、先行系の計算機では発生したアプリケーションを高速版のプログラムに従って実行開始し、追従系の計算機では発生したアプリケーションプロセスを保存しておき、先行系の計算機から指定した条件（例えば実行終了）を満たした旨の通知があるまで実行を延期することにより、そのアプリケーションプロセスの高速版のプログラムが障害を発生するプログラムバグを含まないことを確認するまで実行を延期する。そして、先行系の計算機に障害が発生したことを検出した場合には、追従系の計算機側に保存してあるアプリケーションプロセスのうち、障害の原因となったアプリケーションプロセス

9

を安全版のプログラムに従って実行し、追従系の計算機がダウンするのを防止する。

【0032】この結果、高速版のプログラムにプログラムバグを含まないアプリケーションプロセスは先行系、追従系の計算機群とも高速版のプログラムに従って両系の計算機群とも高速に実行し、高速版のプログラムにプログラムバグを含むアプリケーションプロセスは追従系の計算機群で安全版のプログラムに従って実行することにより、追従系の計算機群には障害が発生することを防ぐ。

【0033】また、一旦先行系の計算機群に障害が発生した後、追従系の計算機群を改めて先行系と追従系とに分類し直すことにより、常に先行系と追従系の両系の計算機を存在させることができ、次に先行系の計算機がダウンした場合に対処することが可能である。

【0034】第3の発明では、システムを制御するプログラム群は“N版プログラム方式”で作成するのではなく、各システム機能を制御するプログラムが求められる機能と処理性能に関する条件を満たすように設計・製作され、通常はそれらプログラム群が計算機内で稼働している。ここで、あるシステム機能を処理するプログラムにバグが潜在していたために処理途中でプログラム停止となり、システムの一部ないし全部の機能停止に至る可能性がある。

【0035】このことを回避するために、第3の発明では第1のプログラムを起動する際、該プログラムの機能を代替する代替プログラムとしての第2のプログラムのプログラム名リスト情報、さらには該リスト情報に必要に応じてそれら代替プログラムの実行優先度情報を付加して起動することにより、第1のプログラムが内在するバグのために実行を停止していたとしても、このプログラム名リスト情報で示される代替プログラムをOSが起動することにより、第1のプログラムの機能の処理を継続する。すなわち、通常は処理性能のよい第1のプログラムで必要な機能処理を実行するが、そのプログラムが何らかの理由で異常となり、プログラム停止となった場合には、第1のプログラムよりは性能は劣るがバグが潜在する確率のより低い代替プログラムである第2のプログラムに処理を代行させて、システム機能の継続を実現する。

【0036】このようにすることにより、複数版のプログラムを作成・管理することなく、プログラム停止になる前は最適手順で作成された第1のプログラムを稼働させてシステムに求められるミッションを達成することが可能となり、また第1のプログラムが停止した場合は計算機に搭載されているOSがプログラム名リスト情報から代替プログラムである第2のプログラムを起動することで、第1のプログラムよりは性能は劣るがミッションを継続して遂行することが可能となる。

【0037】

10

【実施例】以下、図面を参照しながら実施例を説明する。

【0038】図1には本発明の一実施例に係る多重処理システムのブロック構成図が示されている。

【0039】この多重処理システムは、大きく分けて、計算機1，2，3と、共有メモリ4と、この共有メモリ4と各計算機1，2，3とを結合するバス5とで構成されている。

【0040】各計算機1，2，3は、演算装置11，21，31と、実行中プロセスキュー12，22，32および実行遅延プロセスキュー13，23，33を含むローカルメモリ14，24，34と、タイムスライスを発生して各演算装置に知らせるタイマ15，25，35とを備えている。なお、この例の場合、各計算機のタイムスライスの間隔は同一に設定されている。

【0041】共有メモリ4は、実行状態テーブル41，実行終了プロセステーブル42，障害プロセステーブル43を持ち、これらが全計算機とバス5で結合され、全計算機からアクセスされる。

【0042】このように構成された多重処理システムは図2乃至図4に示す流れ図に従って動作する。

【0043】次に、上記のように構成された多重処理システムの動作を図15（a）に示したタイミングで生成、実行されるアプリケーションプロセスP1～P4が投入された場合について説明する。

【0044】この多重処理システムでは、当初、計算機1が先行系、計算機2と計算機3が追従系に分類されているものとする。また、各キューやテーブルの初期状態は全て空で何も登録されておらず、タイムスライス回数は0から始まるものとする。また、本実施例では指定条件（バグを含んでいないことを後段に知らせる条件）として、アプリケーションプロセスの実行終了を用いている。

【0045】まず、実行状態テーブル41の内容が図5（a）に示す状態にあるものとする。この状態で全計算機1，2，3に対してプロセスP1が投入されたとする。

【0046】プロセスP1が投入されると、計算機1では図2に示す流れ図に従って、実行状態テーブル41の内容、すなわち図5（a）に示す内容から自計算機が先行系であることを確認し（S1）、発生プロセスP1を実行中プロセスキュー12に入れる（S2）。

【0047】これと同時に，計算機2でも図2に示す流れ図に従って、自計算機が追従系であることを確認し（S1）、かつ発生プロセスP1が障害プロセステーブル43にないことを確認し（S3）、終了プロセスキューにもないことを確認し（S4）、発生プロセスP1を実行遅延プロセスキュー23に入れる（S6）。計算機3も計算機2と全く同様に、発生プロセスP1を実行遅延プロセスキュー33に入れる。この結果、各計算機の

キューの内容は図5（b）および（c）のようになる。なお、他のキューは依然として空のままである。

【0048】＜回数1のタイムスライス開始＞ここで、タイムスライスが起こると、計算機1では図3に示す流れ図に従って、現在実行中のプロセスがないことを確認し（S11）、実行状態テーブル41の自計算機1のタイムスライス回数を1増して「1」にし（S12）、自計算機が先行系なので（S13）、実行中プロセスキュー12からプロセスP1を取り出し、これを現在実行中のプロセスとして実行状態テーブル41に登録してプロセスP1を起動する（S21）。

【0049】一方，これと同時に計算機2では、計算機1と同様に図3に示す流れ図に従って、現在実行中のプロセスがないことを確認し（S11）、実行状態テーブル41の自計算機2のタイムスライス回数を1増して「1」にし（S12）、自計算機が追従系であることを確認し（S13）、自計算機のタイムスライス回数「1」より2以上タイムスライス回数の遅れている先行系の計算機がないことを確認し（この時点で計算機1のタイムスライス回数は0または1のはずである）（S14）、終了プロセステーブル42には何も入っていないことを確認し（S19）、実行中プロセスキュー22に何も入っていないので何もしない（S21）。計算機3も計算機2と同様である。この結果、実行状態テーブル41および各キューの内容は、図6（a）～（c）のようになる。

【0050】この回数1のタイムスライスの間に計算機1ではプロセスP1がプロセスP2を生成する。すると、計算機1は図2に示す流れ図に従い、自計算機1が先行系なので（S1）、図6（d）に示すように発生プロセスP2を実行中プロセスキュー12に入れる（S2）。

【0051】＜回数2のタイムスライス開始＞次に，再度タイムスライスが起こると、計算機1では図3に示す流れ図に従い、現在実行中のプロセスP1を実行中プロセスキュー12に入れ（S11）、実行状態テーブル41の自計算機1のタイムスライス回数を1増して「2」にし（S12）、自計算機が先行系なので（S13）、実行中プロセスキュー12からプロセスP2を取り出し，これを現在実行中プロセスとして実行状態テーブル41に登録して起動する（S21）。

【0052】これと同時に，計算機2と計算機3においても回数1のタイムスライスのときと同様の手続きが行われる。この結果、実行状態テーブル41および各キューの内容は、図7（a）～（c）のようになる。

【0053】この回数2のタイムスライスの間に計算機1ではプロセスP2がプロセスP3とP4を生成する。すると、計算機1は図2に示す流れ図に従い、自計算機が先行系なので（S1）、図7（d）に示すように発生プロセスP3，P4を実行中プロセスキュー12に入れる（S2）。

【0054】さらに、プロセスP2が終了するので、計算機1は図4の流れ図に従い、自計算機が先行系なので（S31）、図7（e）に示すようにプロセスP2を実行終了プロセステーブル42に入れる（S32）。

【0055】＜回数3のタイムスライス開始＞次に、再度タイムスライスが起こると、計算機1では図3に示す流れ図に従って、現在実行中のプロセスがないことを確認し（S11）、実行状態テーブル41の自計算機1のタイムスライス回数を1増して「3」にし（S12）、自計算機が先行系なので（S13）、実行中プロセスキュー12から図7(b)に示されるP1を取り出し、これを現在実行中プロセスとして実行状態テーブル41に登録して起動する（S21）。

【0056】これと同時に、計算機2と計算機3においても回数1のタイムスライスのときと同様の手続きが行われる。なお、ステップS19で、実行終了プロセステーブル42に入っているP2が各計算機の実行遅延プロセスキー23，33に入っているか確認されるが、P1しか入っていないので、結局、何も行われない。この結果、実行状態テーブル41および各キューの内容は、図8（a）～（c）のようになる。

【0057】さらに、この回数3のタイムスライスの間に計算機1ではプロセスP1が終了する。計算機1は図4に示す流れ図に従って自計算機が先行系なので（S31）、プロセスP1を図8（d）に示すように実行終了プロセステーブル42に入れる（S32）。

【0058】＜回数4のタイムスライス開始＞次に、再度タイムスライスが起こると、計算機1では図3に示す流れ図に従って、現在実行中のプロセスがないことを確認し（S11）、実行状態テーブル41の自計算機1のタイムスライス回数を1増して「4」にし（S12）、自計算機が先行系なので（S13）、実行中プロセスキュー12から図8（b）に示されるP3を取り出し、これを現在実行中プロセスとして実行状態テーブル41に登録して起動する（S21）。

【0059】これと同時に、計算機2では図3の流れ図に従って、現在実行中のプロセスがないことを確認し（S11）、実行状態テーブル41の自計算機2のタイムスライス回数を1増して「4」にし（S12）、自計算機が追従系で（S13）、自計算機のタイムスライス回数「4」より2以上タイムスライス回数の遅れている先行系の計算機がないことを確認し（この時点で計算機1のタイムスライス回数は3または4のはずである）（S14）、図8（d）に示されるように実行終了プロセステーブル42に入っているプロセスP1が図8（c）に示されるように自計算機の実行遅延プロセスキュー23に入っているので（S19）、これを実行中プロセスキュー22に移し（S20）、さらに実行中プロセスキュー22にはP1しか入っていないので，これを

取り出して現在実行中プロセスとして実行状態テーブル４１に登録して起動する（Ｓ２１）。

【００６０】計算機３も計算機２と同様である。この結果、実行状態テーブル４１および各キューの内容は、図９（ａ）～（ｄ）のようになる。

【００６１】この回数４のタイムスライスの間にプロセスＰ３のバグにより計算機１に障害が発生し、計算機１がダウンしたとする。従って、以降、計算機１での処理は中断する。

【００６２】一方、計算機２で実行中のプロセスＰ１がプロセスＰ２を発生し、計算機２は図２に示す流れ図に従って、自計算機が追従系であり（Ｓ１）、発生プロセスＰ２が障害プロセステーブル４３にない（依然空である）ことを確認し（Ｓ３）、実行終了プロセステーブル４２に図８（ｄ）に示すようにＰ２があることを確認し（Ｓ４）、自計算機の実行中プロセスキュー２２に図９(e) に示すように登録する（Ｓ５）。計算機３も同様の動作を行う。

【００６３】＜回数５のタイムスライス開始＞次に、再度タイムスライスが起こると、計算機１は既にダウンしているので何もしない。

【００６４】計算機２では図３に示す流れ図に従って、現在実行中のプロセスＰ１を実行中プロセスキュー２２に入れ（Ｓ１１）、実行状態テーブル４１の自計算機２のタイムスライス回数を１増して「５」にし（Ｓ１２）、自計算機が追従系で（Ｓ１３）、自計算機のタイムスライス回数「５」より２以上タイムスライス回数の遅れている先行系の計算機がないことを確認し（この時点で計算機１のタイムスライス回数は４である）（Ｓ１４）、実行遅延プロセスキュー２３も図９（ｄ）に示すように空なので（Ｓ１９）、図９(e) に示すように実行中プロセスキュー２２に入っているＰ２を取り出し、これを現在実行中プロセスとして実行状態テーブル４１に登録して起動する（Ｓ２１）。

【００６５】計算機３も計算機２と同様である。この結果、実行状態テーブル４１および各キューの内容は、図１０（ａ）（ｂ）に示すようになる（実行遅延プロセスキューは空になる）。

【００６６】この回数５のタイムスライスの間に計算機２，３で実行中のプロセスＰ２がプロセスＰ３，４を生成する。すると、計算機２は図２に示す流れ図に従って、自計算機が追従系で（Ｓ１）、発生プロセスＰ３，Ｐ４が障害プロセステーブル４３にはなく（Ｓ３）、実行終了プロセステーブル４２にもないので（Ｓ４）、図１０（ｃ）に示すようにＰ３，Ｐ４を自計算機の実行遅延プロセスキュー２３に入れる（Ｓ６）。計算機３も同様である。

【００６７】＜回数６のタイムスライス開始＞次に、再度タイムスライスが起こると、計算機２は図３に示す流れ図に従って、現在実行中のプロセスＰ２を実行中プロセスキュー２２に入れ（Ｓ１１）、実行状態テーブル４１の自計算機２のタイムスライス回数を１増して「６」にし（Ｓ１２）、自計算機が追従系で（Ｓ１３）、自計算機のタイムスライス回数「６」より２以上タイムスライス回数の遅れている先行系の計算機１（この時点で依然「４」である）があることを確認し（Ｓ１４）、実行状態テーブル４１において計算機１を「ダウン中」にし、この計算機１が現在実行中となっているプロセスＰ３を図１０（ｃ）に示される実行遅延プロセスキュー２３から削除し、さらに障害プロセステーブル４３に追加する（Ｓ１５）。

【００６８】さらに、自計算機２のサイト番号「２」が追従系の計算機２，３の間で最も小さいので（Ｓ１６）、実行遅延プロセスキュー２３内のプロセスＰ４を実行中プロセスキュー２２に移し（Ｓ１７）、自計算機を先行系にして実行状態テーブル４１を書き換える（Ｓ１８）。

【００６９】引き続き，図８（ｄ）に示されるような実行終了プロセステーブル４３にあるプロセスＰ１，Ｐ２が実行遅延プロセスキュー２３にはないので（Ｓ１９）、実行中プロセスキュー２２からＰ１を取り出し，これ現在実行中プロセスとして実行状態テーブル４１に登録して起動する（Ｓ２１）。

【００７０】計算機３もステップＳ１６～ステップＳ１８以外は計算機２と同様である。すなわち、計算機３のサイト番号「３」は追従系の中で最小ではないので、ステップＳ１７～ステップＳ１８は実行されない。これらの結果、実行状態テーブル４１、障害プロセステーブル４３および各キューの内容は、図１１（ａ）～図１１（ｆ）のようになる。

【００７１】この回数６のタイムスライスの間に、計算機２，３で実行中のプロセスＰ１が終了する。計算機２は図４に示す流れ図に従い、自計算機２が先行系なので（Ｓ３１）、終了プロセスＰ１を実行終了テーブル４２に追加する（Ｓ３２）。ただし、既にＰ１がテーブルにあるので追加しても変化はない。このとき、計算機３は追従系なので何もしない。

【００７２】＜回数７のタイムスライス開始＞次に、再度タイムスライスが起こると、計算機２は図３に示す流れ図にしたがて、現在実行中のプロセスＰ１を実行中プロセスキュー２２に入れ（Ｓ１１）、実行状態テーブル４１の自計算機２のタイムスライス回数を１増して「７」にし（Ｓ１２）、自計算機が先行系なので（Ｓ１３）、図１１（ｃ）に示されるように実行中プロセスキュー２２に入っているＰ４を取り出し、これを現在実行中プロセスとして実行状態テーブル４１に登録して起動する（Ｓ２１）。

【００７３】これと同時に、計算機３は図３に示す流れ図に従って、現在実行中のプロセスＰ１を実行中プロセスキュー３２に入れ（Ｓ１１）、実行状態テーブル４１

15

の自計算機3の部分のタイムスライス回数を1増して「7」にし（S12）、自計算機が追従系で（S13）、自計算機のタイムスライス回数「7」より2以上タイムスライス回数の遅れている先行系の計算機がないことを確認し（S14）、実行終了プロセステーブル42に図8（d）に示されるように入っているプロセスP1，P2が図11（e）に示されるように実行遅延プロセスキュー33にはなく（S19）、図11（f）に示されるように実行中プロセスキュー32も空なので何もしない（S21）。これらの結果、実行状態テーブル41および各キューの内容は、図12（a）～（c）のようになる。

【0074】この回数7のタイムスライスの間に計算機2で実行中のプロセスP4が終了する。計算機2は図4に示す流れ図に従って、自計算機2が先行系なので（S31）、終了プロセスP4を図12（d）に示すように実行終了テーブル42に追加する（S32）。

【0075】<回数8のタイムスライス開始>次に、再度タイムスライスが起こると、計算機2は図3に示される流れ図に従い、現在実行中のプロセスがないことを確認し（S11）、実行状態テーブル41の自計算機2のタイムスライス回数を1増して「8」にする（S12）が、自計算機が先行系で（S13）、図12（b）に示すように実行中プロセスキュー22が空なので何もしない（S21）。

【0076】これと同時に、計算機3は図3に示される流れ図に従い、現在実行中のプロセスがないことを確認し（S11）、実行状態テーブル41の自計算機3のタイムスライス回数を1増して「8」にし（S12）、自計算機が追従系で（S13）、自計算機のタイムスライス回数「8」より2以上タイムスライス回数の遅れている先行系の計算機がないことを確認し（S14）、実行終了プロセステーブル42に入っている図12(d)に示されるようなプロセスP4が図12（c）示されるように自計算機2の実行遅延プロセスキュー33に入っているので（S19）、これを実行中プロセスキュー32に移すが（S21）、実行中プロセスキュー32にはP4しか入っていないので、P4を取り出し、これを現在実行中プロセスとして実行状態テーブル41に登録して起動する（S21）。これらの結果、実行状態テーブル41および各キューの内容は、図13（a）～（c）のようになる。

【0077】この回数8のタイムスライスの間に計算機3で実行中のプロセスP4が終了する。計算機3は追従系なので、図4のステップS31では何もしない。

【0078】ここまでのタイムスライス毎の各計算機で実行されたプロセスの一覧を図14に示す。

【0079】本実施例に係る多重処理システムでは、以上の手順で動作するので、従来の並列多重システムでは実行されなかったプロセスP4を実行させることができる。

16

【0080】なお、上述した実施例では、追従系のアプリケーションプロセスの実行を抑制する手段として、実行遅延プロセスキュー13，23，33を用いているが、このようなキューを用いずに、追従系の計算機上で実行されるアプリケーションプロセス自身が先行系の計算機からの通知を待つルーチンを実行することによっても本実施例とまったく同じ処理の流れが得られる。また、上述した実施例では、先行系の計算機がダウンしたとき、追従系の計算機を新先行系と新追従系とに再構成しているが、再構成することなく、追従系をそのまま先行系にして処理を続行せることもできる。

【0081】次に、第2の発明に係る実施例について説明する。図16は、本発明の他の実施例に係る多重処理システムのブロック構成図である。図1と相対応する部分に同一符号を付して相違点を中心に説明する。

【0082】本実施例においては、各計算機1，2，3内のローカルメモリ14，24，34内に、実行中プロセスキュー12，22，32および実行遅延プロセスキュー13，23，33に加えて、アプリケーションの高速版プログラムを格納する高速版プログラム格納エリア15，25，35およびアプリケーションの安全版プログラムを格納する安全版プログラム格納エリア16，26，36が設けられている点が図1の実施例と異なる。

【0083】次に、上記のように構成された多重処理システムの動作を図17～図19に示す流れ図を参照して説明する。なお、以下の説明では各アプリケーションプロセスP1～P3の高速版のプログラムにバグが無ければ図20（a）に示したタイミングでP1～P3が生成され、実行されるべきところ、実際にはP2の高速版プログラムにバグがある場合の動きについて述べる。また、それぞれのアプリケーションプロセスの安全版プログラムに従った実行は、高速版プログラムに従った実行よりも2倍の時間を要するものとする。比較のため、全アプリケーションプロセスを安全版プログラムに従って実行した場合のタイミングを図20（b）に示す。

【0084】この実施例の多重処理システムでは、当初、計算機1が先行系、計算機2と計算機3が追従系に分類されているものとする。また、各キューやテーブルの初期状態は空で何も登録されておらず、タイムスライス回数は0から始まるものとする。高信頼版プログラム格納エリア15，25，35にはそれぞれP1～P3の高速版のプログラムが格納され、安全版プログラム格納エリア16，26，36にはそれぞれP1～P3の高速版と同一機能を有する安全版プログラムが格納されている。また、本実施例では指定条件（障害が発生しなかったことを後段に知らせる条件）として、アプリケーションプロセスの実行終了を用いている。

【0085】まず、実行状態テーブル41の内容は図21（a）に示す状態にあるものとする。この状態で全計

算機にプロセスＰ１が投入されたとする。プロセスＰ１が投入されると、計算機１では図２１（ａ）に示す実行状態テーブル４１から自計算機が先行系であることを確認した後、図１７に示す流れ図に従って、まず発生プロセスが障害プロセステーブルにないことを確認し（Ｓ４１）、発生プロセスＰ１を該プロセスのプログラムアドレスを高速版プログラム格納エリア１４に格納されているＰ１の高速版プログラムのアドレスにして、実行中プロセスキュー１２に入れる（Ｓ４２）。

【００８６】一方、これと同時に計算機２ではやはり図１７に示す流れ図に従って、自計算機が追従系で、かつ発生プロセスＰ１が障害プロセステーブル４３に無く、終了プロセスキューにも無いことを確認し（Ｓ４３）、発生プロセスＰ１を実行遅延プロセスキュー２３に入れる（Ｓ４５）。また、計算機３も計算機２と全く同様にして、プロセスＰ１を実行遅延プロセスキュー３３に入れる。この結果、各計算機１，２，３のキューの内容は図２１（ｂ）（ｃ）のようになる。

【００８７】なお、以下の説明で用いる図２１～図３３では、実行中キューに格納されているアプリケーションプロセスをそのプログラムアドレスとして高速版プログラムのアドレスになっているプロセスをＰｎ（高）で表記し、同様に安全版プログラムのアドレスになっているプロセスをＰｎ（安）で表記する。但しｎ＝１～３である。他のキューは、以前空のままであるとする。

【００８８】＜回数１のタイムスライス開始＞ここで、タイムスライスが起きる。すると計算機１では図１８に示す流れ図に従って、現在実行中のプロセスが無いことを確認して何もせず（Ｓ５１）、実行状態テーブル４１の自計算機１の部分のタイムスライス回数を１増して「１」にし（Ｓ５２）、自計算機が先行系であることを確認し（Ｓ５３）、実行中プロセスキュー１２からプロセスＰ１を取り出してこれを現在実行中のプロセスとして実行状態テーブル４１に登録してプロセスＰ１を起動する（Ｓ６１）。このＰ１はプログラムアドレスが高速版プログラムのものなので、高速版プログラムに従って実行される。

【００８９】一方、これと同時に計算機２では、計算機１と同様に図１８に示す流れ図に従って、現在実行中のプロセスが無いのでＳ５１では何もせず、実行状態テーブル４１の自計算機２の部分のタイムスライス回数を１増して「１」にし（Ｓ５２）、自計算機が追従系であることを確認し（Ｓ５３）、自計算機の部分のタイムスライス回数「１」より２以上タイムスライス回数が遅れている先行系の計算機がないことを確認し（計算機１の部分のタイムスライス回数はこの時点で０または１のはずである）（Ｓ５４）、終了プロセスキュー４２には何も入っていないことを確認し（Ｓ５９）、さらに実行中プロセスキュー２２にも何も入っていないので何もしない（Ｓ５１）。計算機３も計算機２と同様である。この結果、実行状態テーブル４１および各キューの内容は図２２（ａ）～（ｃ）のようになる。

【００９０】この回数１のタイムスライスの間に計算機１ではプロセスＰ１がプロセスＰ２を生成する。すると、計算機１は図１７に示す流れ図に従い発生プロセスＰ２が依然空のままである障害プロセステーブル４３にないことを確認し（Ｓ４１）、さらに自計算機が先行系であることを確認して（Ｓ４３）、発生プロセスＰ２を該プロセスのプログラムアドレスを高速版プログラム格納エリア１４に格納されているＰ２の高速版プログラムのアドレスにして、実行中プロセスキュー１２に入れる（Ｓ４４）。この様子を図２２（ｄ）に示す。

【００９１】＜回数２のタイムスライス開始＞次に再度タイムスライスが起きると、計算機１では図１８に示す流れ図に従い、現在実行中のプロセスＰ１を実行中プロセスキュー１２に入れ（Ｓ５１）、実行状態テーブル４１の自計算機１の部分のタイムスライス回数を１増して「２」にし（Ｓ５２）、自計算機が先行系であることを確認し（Ｓ５３）、図２３（ａ）中に示す実行中プロセスキュー１２からＰ２を取り出して、これを現在実行中プロセスとして実行状態テーブル４１に登録して起動する（Ｓ６１）。このＰ２はプログラムアドレスが高速版プログラムのものなので、高速版プログラムに従って実行される。これと同時に、計算機２と計算機３では、やはり回数１のタイムスライスの時と同様の手続きが行われる。この結果、実行状態テーブル４１および各キューの内容は図２３（ａ）～（ｃ）のようになる。

【００９２】この回数２のタイムスライスの間に計算機１では高速版プログラムに従ったプロセスＰ２がプログラムのバグで障害を発生し、計算機１がダウンする。

【００９３】＜回数３のタイムスライス開始＞次に、再度タイムスライスが起きると、計算機１は既にダウンしているので何もせず、計算機２では図１８に示す流れ図に従って、現在実行中のプロセスは無いのでＳ５１では何もせず、実行状態テーブル４１の自計算機２の部分のタイムスライス回数を１増して「３」にし（Ｓ５２）、自計算機が追従系で（Ｓ５３）、自計算機の部分のタイムスライス回数「３」より２以上タイムスライス回数が遅れている先行系の計算機は無く（計算機１のタイムスライス回数は、この時点で２である）（Ｓ５）、実行遅延プロセスキュー２３が空で（Ｓ５）、さらに実行中プロセスキューにも何も入っていないので、Ｓ６１でも何もしない。この結果、実行状態テーブル４１および各キューの内容は、図２４（ａ）～（ｃ）のようになる。

【００９４】＜回数４のタイムスライス開始＞次に、再度タイムスライスが起きると、計算機２は図１８に示す流れ図に従って、現在実行中のプロセスが無いのでＳ５１は何もせず、実行状態テーブル４１の自計算機２の部分のタイムスライス回数を１増して「４」にし（Ｓ５２）、自計算機が追従系で（Ｓ５３）、自計算機のタイ

19

ムスライス回数「4」より2以上タイムスライス回数が遅れている先行系の計算機1（この時点で依然「2」である）があるので（S54）、実行状態テーブル41に於いて該計算機1を「ダウン中」にし、該計算機1が現在実行中となっているプロセスP2を、障害プロセステーブルに加える。なお、図24（c）に示す実行遅延プロセスキュー23にはP2は無いので、実行遅延キューから実行中キューへ移す処理は無い）（S55）。

【0095】さらに、自計算機2のサイト番号「2」は追従系計算機2，3の間で最も小さいので（S56）、図24（c）に示す実行遅延プロセスキュー23のプロセスP1を取り出し、該プロセスのプログラムアドレスを高速版プログラム格納エリア26に格納されているP1の高速版プログラムのアドレスにして、実行中プロセスキュー22に入れ（S57）、自計算機を先行系にして実行状態テーブル41を書き換える（S58）。そして、実行終了プロセステーブル42は依然空なので（S59）、実行中プロセスキュー22からP1を取り出して実行状態テーブル41に登録し、起動する（S61）。このP1はプログラムアドレスが高速版プログラムのものなので、高速版プログラムに従って実行される。

【0096】一方、計算機3では、図18に示す流れ図に従って、S55までは計算機2と全く同じに動作するが、自計算機3のサイト番号「3」は追従系計算機2，3の間で最小ではなく（S56）、実行終了プロセステーブル42は依然空で（S59）、さらに実行中プロセスキュー32も空なので、S61でも何もしない。この結果、各計算機1，2，3の各キューおよびテーブルは、図25（a）～（e）のようになる。

【0097】この回数4のタイムスライスの間に、計算機2ではプロセスP1がプロセスP2を生成する。すると、計算機2は図17に示す流れ図に従って、発生プロセスP2が図25図（e）に示す障害プロセステーブルに存在するので、該プロセスのプログラムアドレスを安全版プログラム格納エリア25に格納されているP2の安全版プログラムのアドレスにして、実行中プロセスキュー22に入れる（S42）。この結果、計算機2の実行中プロセスキューは図25（f）のようになる。

【0098】<回数5のタイムスライス開始>次に、再度タイムスライスが起きると、計算機2は図18に示す流れ図に従って、現在実行中のプロセスP1を実行中プロセスキューに入れ（S51）、実行状態テーブル41の自計算機2のタイムスライス回数を1増して「5」にし（S52）、自計算機が先行系なので（S53）、図26中の実行中プロセスキュー22からP2を取り出して、これを現在実行中プロセスとして実行状態テーブル41に登録して起動する（S61）。このP2はプログラムアドレスが安全版プログラムのものなので、安全版プログラムに従って実行される。

20

【0099】これと同時に、計算機3は図18に示す流れ図に従って、現在実行中のプロセスは無いのでS51では何もせず、実行状態テーブル41の自計算機3の部分のタイムスライス回数を1増して「5」にし（S52）、自計算機が追従系で（S53）、自計算機のタイムスライス回数「5」より2以上タイムスライス回数が遅れている先行系の計算機は無く（S54）、終了プロセスプロセスキュー42が空で（S59）、さらに実行中プロセスキューにも何も入っていないのでS61でも何もしない。この結果、実行状態テーブル41および各キューの内容は、図26（a）～（f）のようになる。

【0100】この回数5のタイムスライスの間に、計算機2ではプロセスP2がプロセスP3を生成する。すると、計算機2は図17に示す流れ図に従って、発生プロセスP3が障害プロセステーブル43に無く（S41）、自計算機が先行系なので（S43）、発生プロセスP3を該プロセスのプログラムアドレスを高速版プログラム格納エリア24に格納されているP3の高速版プログラムのアドレスにして、実行中プロセスキュー22に入れる（S44）。この様子を図26図（f）に示す。

【0101】<回数6のタイムスライス開始>次に、再度タイムスライスが起きると、計算機2は図18に示す流れ図に従って、現在実行中のプロセスP2を実行中プロセスキューに入れ（S51）、実行状態テーブル41の自計算機2の部分のタイムスライス回数を1増して「6」にし（S52）、自計算機が先行系なので（S53）、図27（d）に示す実行中プロセスキュー22からP1を取り出してこれを現在実行中プロセスとして実行状態テーブル41に登録して起動する（S61）。このP1はプログラムアドレスが高速版プログラムのものなので高速版プログラムに従って実行される。計算機3の動作は、タイムスライス回数5の時と全く同じ動作である。

【0102】この結果、実行状態テーブル41および各キューの内容は、図27（a）～（d）のようになる。

【0103】この回数6のタイムスライスの間に、計算機2で実行中のプロセスP1は終了するので、計算機2は図19に示す流れ図に従い、自計算機2が先行系であることを確認し（S71）、終了プロセスP1を実行終了テーブル42に追加する（S72）。この様子を図27（e）に示す。

【0104】<回数7のタイムスライス開始>次に、再度タイムスライスが起きると、計算機2は図18に示す流れ図に従って、現在実行中のプロセスが無い（P1はタイムスライスの途中で終了している）のでS51では何もせず、実行状態テーブル41の自計算機2の部分のタイムスライス回数を1増して「7」にし（S52）、自計算機が先行系なので（S53）、図27（b）に示す実行中プロセスキュー22からプロセスP3を取り出

して、これを現在実行中プロセスとして実行状態テーブル４１に登録して起動する（Ｓ６１）。このＰ３はプログラムアドレスが高速版プログラムのものなので、高速版プログラムに従って実行される。

【０１０５】一方、計算機３は図１８に示す流れ図に従って、現在実行中のプロセスが無いのでＳ５１では何もせず、実行状態テーブル４１の自計算機３の部分のタイムスライス回数を１増して「7」にし（Ｓ５２）、自計算機が追従系で（Ｓ５３）、自計算機のタイムスライス回数「7」より２以上タイムスライス回数が遅れている先行系の計算機は無いが（Ｓ５４）、図２７（ｅ）に示す終了プロセスプロセスキュー４２にあるプロセスＰ１が図２７図（ｄ）に示す実行遅延プロセスキュー３３にあるので（Ｓ５９）、該プロセスのプログラムアドレスを高速版プログラム格納エリア３４に格納されているＰ１の高速版プログラムのアドレスにして、実行中プロセスキュー３２に入れる（Ｓ６０）。さらに、実行中プロセスキュー３２から今入れたプロセスＰ１を取り出して、これを現在実行中のプロセスとして実行状態テーブル４１に登録してプロセスＰ１を起動する（Ｓ６１）。このＰ１はプログラムアドレスが高速版プログラムのものなので、高速版プログラムに従って実行される。

【０１０６】この結果、実行状態テーブル４１および各キューの内容は、図２８（ａ）～（ｄ）のようになる。

【０１０７】この回数７のタイムスライスの間に計算機２で実行中のプロセスＰ３は終了するので、計算機２は図１９に示す流れ図に従い、自計算機２が先行系なので（Ｓ７１）、終了プロセスＰ３を実行終了テーブル４２に追加する（Ｓ７２）。この様子を図２８（ｅ）に示す。

【０１０８】一方、この回数７のタイムスライスの間に、計算機３ではプロセスＰ１がプロセスＰ２を生成する。すると、計算機３は図１８に示す流れ図に従い発生プロセスＰ２が図２６（ｅ）に示す障害プロセステーブルに存在するので（Ｓ４１）、該プロセスのプログラムアドレスを安全版プログラム格納エリア３５に格納されているＰ２の安全版プログラムのアドレスにして、実行中プロセスキュー３２に入れる（Ｓ４２）。この結果、計算機３の実行中プロセスキューは図２８（ｆ）のようになる。

【０１０９】＜回数８のタイムスライス開始＞次に、再度タイムスライスが起きると、計算機２は図１８に示す流れ図に従って、現在実行中のプロセスが無い（Ｐ３はタイムスライスの途中で終了している）のでＳ５１では何もせず、実行状態テーブル４１の自計算機２の部分のタイムスライス回数を１増して「8」にし（Ｓ５２）、自計算機が先行系なので（Ｓ５３）、実行中プロセスキュー２２（図３６）からプロセスＰ２を取り出してこれを現在実行中プロセスとして実行状態テーブル４１に登録して起動する（Ｓ６１）。このＰ２はプログラムアドレスが安全版プログラムのものなので安全版プログラムに従って実行される。

【０１１０】一方、計算機３は図１８に示す流れ図に従って、現在実行中のプロセスＰ１を実行中キュー３２に入れ（Ｓ５１）、実行状態テーブル４１の自計算機３の部分のタイムスライス回数を１増して「8」にし（Ｓ５２）、自計算機が追従系で（Ｓ５３）、自計算機のタイムスライス回数「8」より２以上タイムスライス回数が遅れている先行系の計算機は無く（Ｓ５４）、図２７図（ｅ）に示す実行終了プロセスプロセスキュー４２にあるプロセスＰ１，Ｐ３は図２７（ｄ）に示す実行遅延プロセスキューに無いので（Ｓ５９）、実行中プロセスキュー３２からプロセスＰ２を取り出して、これを現在実行中のプロセスとして実行状態テーブル４１に登録して起動する（Ｓ６１）。このＰ２はプログラムアドレスが安全版プログラムのものなので、安全版プログラムに従って実行される。

【０１１１】この結果、実行状態テーブル４１および各キューの内容は、図２９（ａ）～（ｄ）のようになる。

【０１１２】この回数８のタイムスライスの間に計算機２で実行中のプロセスＰ３は終了するので、計算機２は図１９に示す流れ図に従い自計算機２が先行系なので（ステップ７１）、終了プロセスＰ２を実行終了テーブル４２に追加する（ステップ７２）。この様子を図２９（ｅ）に示す。

【０１１３】一方、この回数８のタイムスライスの間に計算機３ではプロセスＰ２がプロセスＰ３を生成する。すると、計算機３は図１８に示す流れ図に従い、発生プロセスＰ３が図２６（ｅ）に示す障害プロセステーブルに存在せず（Ｓ４１）、発生プロセスＰ３が実行終了プロセステーブル４２にあるので（Ｓ４３）、該プロセスのプログラムアドレスを高速版プログラム格納エリア３６に格納されているＰ３の高速版プログラムのアドレスにして、実行中プロセスキュー３２に入れる（Ｓ４２）。この結果、計算機３の実行中プロセスキューは図２９（ｆ）に示すようになる。

【０１１４】＜回数９のタイムスライス開始＞次に、再度タイムスライスが起きると、計算機２は図１８に示す流れ図に従って、現在実行中のプロセスが無い（Ｐ２はタイムスライスの途中で終了している）のでＳ５１では何もせず、実行状態テーブル４１の自計算機２の部分のタイムスライス回数を１増して「9」にし（Ｓ５２）、自計算機が先行系で（Ｓ５３）、図２９（ｂ）に示す実行中プロセスキュー２２が空であるのでＳ６１も何もしない。これ以降、計算機２は毎回のタイムスライス開始時に図１８に示す流れ図に従ってＳ５２でタイムスライス回数を１づつ増す以外は何もしないので、以降動きの説明を省略する。

【０１１５】一方、計算機３は図１８に示す流れ図に従って、現在実行中のプロセスＰ２を実行中キュー３２に

入れ（S51）、実行状態テーブル41の自計算機3の部分のタイムスライス回数を1増して「9」にし（S52）、自計算機が追従系で（S53）、自計算機のタイムスライス回数「9」より2以上タイムスライス回数が遅れている先行系の計算機は無く（S54）、実行遅延プロセスキュー33も空なので実行中プロセスキュー32からプロセスP1を取り出してこれを現在実行中のプロセスとして実行状態テーブル41に登録して起動する（S61）。このP1はプログラムアドレスが安全版プログラムのものなので、高速版プログラムに従って実行される。

【0116】この結果、実行状態テーブル41および各キューの内容は、図30（a）～（c）のようになる。

【0117】この回数9のタイムスライスの間に計算機3で実行中のプロセスP1は終了するが、計算機3は図19に示す流れ図に従って自計算機3が追従系なので何もしない（S71）。

【0118】＜回数10のタイムスライス開始＞次に、再度タイムスライスが起きると、計算機3は図18に示す流れ図に従って、現在実行中のプロセスが無いので（P1は回数9のタイムスライス間に終了している）S51では何もせず、実行状態テーブル41の自計算機3の部分のタイムスライス回数を1増して「10」にし（S52）、自計算機が追従系で（S53）、自計算機のタイムスライス回数「10」より2以上タイムスライス回数が遅れている先行系の計算機は無く（S54）、実行遅延プロセスキュー33も空なので、実行中プロセスキュー32からプロセスP3を取り出してこれを現在実行中のプロセスとして実行状態テーブル41に登録して起動する（S61）。このP3はプログラムアドレスが高速版プログラムのものなので、高速版プログラムに従って実行される。

【0119】この結果、実行状態テーブル41および各キューの内容は、図31（a）～（c）のようになる。

【0120】この回数10のタイムスライスの間に、計算機3で実行中のプロセスP3は終了するが、計算機3は図19に示す流れ図に従って自計算機3が追従系なので何もしない（S71）。

【0121】＜回数11のタイムスライス開始＞次に、再度タイムスライスが起きると、計算機3は図18に示す流れ図に従って、現在実行中のプロセスが無いので（P3は回数10のタイムスライス間に終了している）S51では何もせず、実行状態テーブル41の自計算機3の部分のタイムスライス回数を1増して「10」にし（S52）、自計算機が追従系で（S53）、自計算機のタイムスライス回数「11」より2以上タイムスライス回数が遅れている先行系の計算機は無く（S54）、実行遅延プロセスキュー33も空なので実行中プロセスキュー32からプロセスP2を取り出してこれを現在実行中のプロセスとして実行状態テーブル41に登録して

起動する（S61）。このP2はプログラムアドレスが安全版プログラムのものなので、高速版プログラムに従って実行される。

【0122】この結果、実行状態テーブル41および各キューの内容は、図32（a）～（b）のようになる。

【0123】この回数11のタイムスライスの間に、計算機2で実行中のプロセスP2は終了するが、計算機3は図19に示す流れ図に従って自計算機3が追従系なので何もしない（S71）。

【0124】ここまでのタイムスライス毎の各計算機で実行されたプロセスの一覧を図334に示す。

【0125】以上の操作で、本実施例のシステムによれば生成されたアプリケーションプロセスP2の高速版プログラムにプログラムバグが含まれても、システムが止まること無く処理が続けられ、また先行系、追従系の計算機ともプログラムにプログラムバグの含まれないアプリケーションプロセスP1，P3は高速版のプログラムに従って実行され、システム全体の処理時間が遅くなることは無いことがわかる。

【0126】このように本実施例によれば、アプリケーションプロセスに対し、高速版のプログラムと、時間はかかるがプログラムバグの無い安全版のプログラムを用意することにより、高速版のプログラムにプログラムバグが含まれるアプリケーションプロセス以外は高速版だけの場合と同等の処理時間で、処理が中断することを避けられる。

【0127】なお、以上説明した第1および第2の発明に係る実施例については、図34および図35に示すような変形が可能である。図34に示す実施例は、図1または図16に示した多重処理システムにおいて、計算機1，2，3にコピー装置18，28，38cをそれぞれ付設し、実行状態テーブル41、実行終了プロセステーブル42および障害プロセステーブル43を各計算機のローカルメモリ14，24，34に搭載し、ローカルメモリ14，24，34の内容（特に、テーブル41，42，43の内容）を随時相互に転送してコピーできるようにしたものである。図35に示す実施例は、図1または図16におけるタイマ15，25，35に代えてバス5に接続した共有タイマ6を用いたものである。

【0128】次に、第3の発明に係る実施例について説明する。図36は、本実施例に係る多重処理システムのブロック構成図である。計算機1，2，3はそれぞれバス5（例えばLAN）で接続されており、各計算機1，2，3内には通信装置101，202，301、演算装置102，202，302、OS格納エリア103，203，303およびプログラム格納エリア105，205，305が設けられ、OS格納エリア103，203，303には、図39に示すようなプログラム名とその開始アドレスの対応を示したプログラム管理テーブル104，204，304が設けられている。

25

【0129】図37は、計算機1内のOS格納エリア103，203，303に格納されたOSにより実行制御されるプログラム格納エリア105，205，305内のプログラムP001，P002，…が稼働している状態を示している。プログラムP001，P002，…間の情報交換はOSが制御して、他計算機のプログラムにデータ転送する場合には、バス5を介してデータの送受信処理を制御する。なお、以後OS格納エリア103に格納されたOSを計算機1内のOS、またOS格納エリア203に格納されたOSを計算機2内のOS、さらにOS格納エリア303に格納されたOSを計算機3内のOSという。

【0130】図38に、各プログラムの処理機能を示す。プログラムP001は、2つのベクトルの和を求め、その絶対値の2乗値を計算するプログラムであり、計算機1で稼働している。プログラムP002は、2つのベクトルの和を計算するプログラムであり、計算機2で稼働している。プログラムP003は、ベクトルの絶対値（大きさ）を計算するプログラムであり、計算機2で稼働している。プログラムP004は、数値（スカラ量）の2乗を計算するプログラムであり、計算機3で稼働している。そして、プログラムP005は、入力される数値の積分値を計算するプログラムであり、計算機3で稼働している。

【0131】ここで、プログラムP001を使って2つのベクトルの和を求め、その絶対値の2乗値を計算したいユーザがキーボード6からプログラムP001を呼び出して、2つのベクトル量を入力するときの処理の流れを図40により説明する。

【0132】まず、ユーザはキーボード6からプログラムP001を呼び出す際に、プログラム名”P001”とともに、実行優先度情報が付加された”P001”の代替プログラム名リスト［P002（1），P003（2），P004（3）］を入力する（S101）。リストに記載されているP002（1）を例に説明すると、P002が代替プログラム名を示し、（1）はプログラムP002の実行優先度が1番であることを示している。

【0133】次に、計算機1内のOSは、キーボード6から入力された起動対象プログラムを示す起動プログラム名“P001”と、その代替プログラムリスト［P002（1），P003（2），P004（3）］を記憶する（S102）。

【0134】次に、計算機1内のOSは、入力された起動プログラム名が自計算機1に登録されているか調べる（S103）。この例の場合、起動プログラム名はプログラム“P001”であり、これが登録されていることが確認されるので、プログラム“P001”を起動する（S105）。起動プログラム名が自計算機1に登録されていなければ、S104に進む。

26

【0135】こうしてプログラム“P001”が起動され、プログラム手順に従って2つのベクトル量の値の入力をユーザに要求する。この要求を受けてユーザがキーボード6からデータを入力すると（S106）、“プログラムP001”は2つのベクトルの和を求め、その絶対値の2乗値を計算して図示しないCRT上に結果の値を出力する（S107）。

【0136】ここで、計算機1で稼働していたプログラム“P001”がプログラム停止したとする。例えば、キーボード6から入力されたデータが数値以外の形式で入力された場合を想定すると、プログラム“P001”には異質入力データに対する例外処理が施されていなかったため、プログラムエラーを生じてOSがプログラムの実行を異常終了として停止したとする。

【0137】この場合、計算機1内のOSは、自計算機1内にプログラム“P001”が登録されていないので、プログラム“P001”が必要とする入力データをユーザからキーボード6を介して受信すると、他の計算機2，3内のOSに対して、プログラム“P001”がプログラム格納エリア205，305に登録されているかを問い合わせる（S104）。この例では、計算機2，3内のプログラム格納エリア205，305にもプログラム“P001”は存在しないので、代替プログラムリスト内のプログラム“P002”をまず起動する（S108）。

【0138】プログラム“P002”の起動は、例えば計算機1内のOSが他の計算機2，3内のOSに対して代替リストと最終代替プログラムからの結果返信先情報およびプログラム“P002”の起動要求と入力データから構成されるメッセージを放送することで実現できる。この例の場合、プログラム“P002”は計算機2内のプログラム格納エリア205に存在しているので、計算機2内のOSは、計算機1内のOSからの起動要求を受けて、プログラム“P002”を起動する。プログラム“P002”は、起動されるとメッセージ中の2つのベクトル量の入力データを読み込んで、2つのベクトルの和を計算し、CRTに出力する。計算機2内のOSは、プログラム“P002”が計算したベクトル和の値が記憶されている記憶領域から該出力値を読み出す。そして、計算機2内のOSは、代替リストからプログラム“P002”を削除し、残りのリスト情報と最終返信先情報およびプログラム“P003”の起動要求とベクトル和の結果をメッセージとして放送する。

【0139】プログラム“P003”は計算機2に存在するので、計算機2内のOSがプログラム“P003”を起動する。プログラム“P003”は、計算機2内のOSからベクトル量を入力されるとその絶対値を計算し、CRTに出力する。計算機2内のOSは、プログラム“P003”が計算したベクトルの絶対値が記憶されている記憶領域から該出力値を読み出し、その値を次の

27

代替プログラムであるプログラム“Ｐ００４”に入力データとして渡す。

【０１４０】すなわち、計算機２内のＯＳは、代替リストからプログラム“Ｐ００３”を削除し、残りのリスト情報とプログラム“Ｐ００４”の起動要求およびベクトルの絶対値（スカラ値）をメッセージとして放送する。プログラム“Ｐ００４”は計算機３に存在しているので、計算機３内のＯＳが“Ｐ００４”を起動する。プログラム“Ｐ００４”は計算機３内のＯＳからスカラ値を入力されると、その２乗を計算しＣＲＴに出力する。プログラム“Ｐ００４”が代替リスト中の最終代替プログラムであるので、メッセージに付加されている返信先情報である計算機１内のＯＳに結果を返信する（Ｓ１１０）。

【０１４１】そして、計算機１内のＯＳは、計算機３内のＯＳより返信されてきたメッセージに含まれている最終結果を自計算機のＣＲＴに出力する（Ｓ１１１）。

【０１４２】このように、本実施例によれば複数の版のプログラムを作成して管理することなく、プログラム停止になる前は最適手順で作成されたプログラム“Ｐ００２”を稼働させてシステムに求められるミッションを達成することが可能となる。そして、プログラム“Ｐ００２”が停止した場合は、他の計算機２，３に搭載されているＯＳが代替プログラムリスト情報から代替プログラムを逐次起動することで、プログラム“Ｐ００２”よりは性能は劣るが、「２つのベクトルの和を求め、その絶対値の２乗値を計算する」という当初のミッションを継続して遂行することが可能となる。

【０１４３】

【発明の効果】以上説明したように、本発明によればハードウェア障害はもとより、プログラムのバグで計算機に障害が発生した場合でも、全計算機がダウンするのを避けてシステム全体の処理を続行することができる。

【０１４４】すなわち、第１の発明によればアプリケーションプロセスのプログラムバグによる障害で計算機がダウンした場合でもシステム全体を停止させることなく、処理を続行させることができる。

【０１４５】また、第２の発明によればアプリケーションプロセスに対し、高速版プログラムと、時間はかかるがプログラムバグの無い安全版プログラムを用意することにより、高速版のプログラムにプログラムバグが含まれるアプリケーションプロセス以外は、安全版プログラムにより高速版プログラムのみの場合と同等の処理時間で処理の中断を避けることができる。

【０１４６】さらに、第３の発明によれば複数版のプログラムを作成・管理することなく、プログラム停止になる前は最適手順で作成された第１のプログラムを稼働させてシステムに求められるミッションを達成することが可能となり、また第１のプログラムが停止した場合は計算機に搭載されているＯＳがプログラム名リスト情報から代替プログラムである第２のプログラムを起動することで、第１のプログラムよりは性能は劣るがミッションを継続して遂行することが可能となる。

28

【図面の簡単な説明】

【図１】第１の発明に係る多重処理システムの実施例を示すブロック構成図

【図２】同実施例のシステムにおけるアプリケーションプロセス発生時の動作を示す流れ図

【図３】同実施例のシステムにおけるタイムスライス開始時の動作を示す流れ図

【図４】同実施例のシステムにおけるアプリケーションプロセス終了時の動作を示す流れ図

【図５】同実施例のシステムの初期状態における実行状態テーブルおよび各キューの内容を示す図

【図６】同実施例のシステムにおける回数１のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図

【図７】同実施例のシステムにおける回数２のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図

【図８】同実施例のシステムにおいて回数３のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図

【図９】同実施例のシステムにおける回数４のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図

【図１０】同実施例のシステムにおける回数５のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図

【図１１】同実施例のシステムにおける回数６のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図

【図１２】同実施例のシステムにおける回数７のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図

【図１３】同実施例のシステムにおける回数８のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図

【図１４】同実施例のシステムにおいて各計算機が実行したプロセスの一覧を示す図

【図１５】従来の並列多重処理システムでアプリケーションプロセスを実行させたときの期待される実行例（ａ）とバグによりマシンがダウンした例（ｂ）を示す図

【図１６】第２の発明に係る多重処理システムの実施例を示すブロック構成図

【図１７】同実施例のシステムにおけるアプリケーションプロセス発生時の動作を示す流れ図

【図１８】同実施例のシステムにおけるタイムスライス開始時の動作を示す流れ図

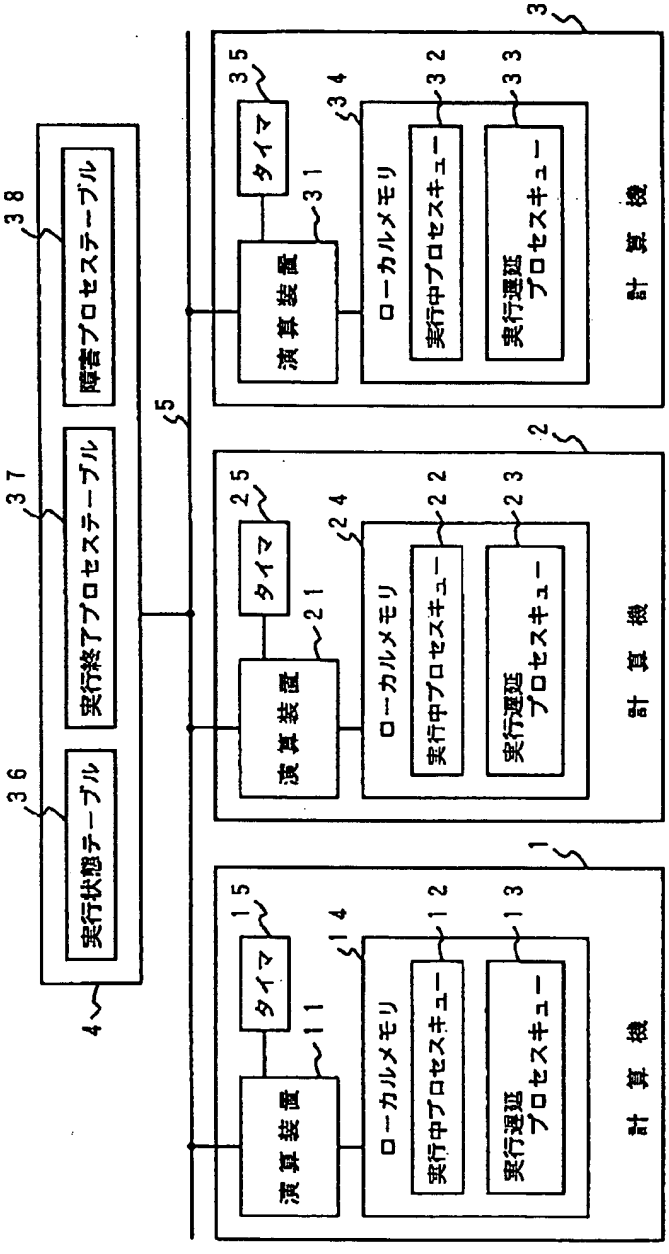
29

【図１９】同実施例のシステムにおけるアプリケーションプロセス終了時の動作を示す流れ図
【図２０】同実施例のシステムでアプリケーションプロセスを実行させたときの期待される実行例（ａ）と高速版プログラムにバグがある場合の例（ｂ）を示す図
【図２１】同実施例のシステムの初期状態における実行状態テーブルおよび各キューの内容を示す図
【図２２】同実施例のシステムにおける回数１のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図
【図２３】同実施例のシステムにおける回数２のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図
【図２４】同実施例のシステムにおける回数３のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図
【図２５】同実施例のシステムにおける回数４のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図
【図２６】同実施例のシステムにおける回数５のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図
【図２７】同実施例のシステムにおける回数６のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図
【図２８】同実施例のシステムにおける回数７のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図
【図２９】同実施例のシステムにおける回数８のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図
【図３０】同実施例のシステムにおける回数９のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図
【図３１】同実施例のシステムにおける回数１０のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図
【図３２】同実施例のシステムにおける回数１１のタイムスライス開始時処理後の実行状態テーブルおよび各キューの内容を示す図

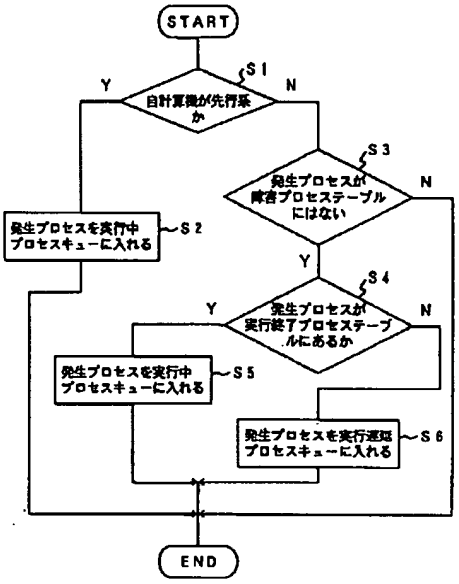
30

【図３３】同実施例のシステムにおいて各計算機が実行したプロセスの一覧を示す図
【図３４】第２の発明に係る多重処理システムの他の実施例を示すブロック構成図
【図３５】第２の発明に係る多重処理システムの別の実施例を示すブロック構成図
【図３６】第３の発明に係る多重処理システムの一実施例を示すブロック構成図
【図３７】同実施例のシステムにおける各計算機内に格納されたプログラム名を示す図
【図３８】同実施例のシステムにおける各プログラムの処理例を示す図
【図３９】同実施例のシステムにおけるプログラム管理テーブルの内容を示す図
【図４０】同実施例のシステムにおける処理手順を示す流れ図
【符号の説明】
１，２，３…計算機
４…共有メモリ
５…バス
６…共有タイマ
７…キーボード
１１，２１，３１…演算装置
１２，２２，３２…実行中プロセスキュー
１３，２３，３３…実行遅延プロセスキュー
１４，２４，３４…ローカルメモリ
１５，２５，３５…タイマ
１６，２６，３６…高速版プログラム格納エリア
１７，２７，３７…安全版プログラム格納エリア
１８，２８，３８…コピー装置
４１…実行状態テーブル
４２…実行終了プロセステーブル
４３…障害プロセステーブル
１０１，２０１，３０１…通信装置
１０２，２０２，３０２…演算装置
１０３，２０３，３０３…ＯＳ格納エリア
１０４，２０４，３０４…プログラム管理テーブル
１０５，２０５，３０５…プログラム格納エリア

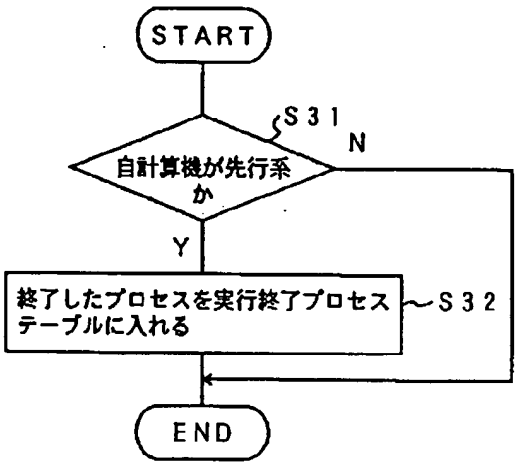
【図1】


計算機
演算装置
タイマ
ローカルメモリ
実行中プロセスキュー
実行遅延プロセスキュー
実行状態テーブル
実行終了プロセステーブル
障害プロセステーブル
1
2
3
4
5
11
12
13
14
15
21
22
23
24
25
31
32
33
34
35
36
37
38

【図2】



【図4】



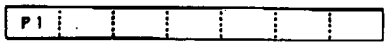
【図5】

(a)

計算機サイト番号	1	2	3
先行系／追従系	先行系	追従系	追従系
タイムスライス回数	0	0	0
現在実行中プロセス	－	－	－

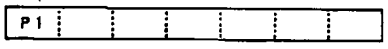
実行状態テーブルの初期状態

(b)



プロセスP1投入後の計算機1の実行中プロセスキュー

(c)



プロセスP1投入後の計算機2，3の実行遅延プロセスキュー

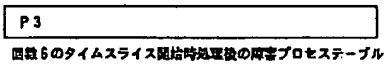
【図11】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン中	先行系	追従系
タイムスライス回数	4	6	6
現在実行中プロセス	P3	P1	P1

回数6のタイムスライス開始時処理後の実行状態テーブル

(b)

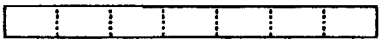


回数6のタイムスライス開始時処理後の障害プロセステーブル

(c)

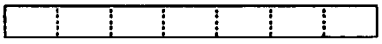
回数6のタイムスライス開始時処理後の計算機2の実行中プロセスキュー

(d)



回数6のタイムスライス開始時処理後の計算機2の実行遅延プロセスキュー

(e)

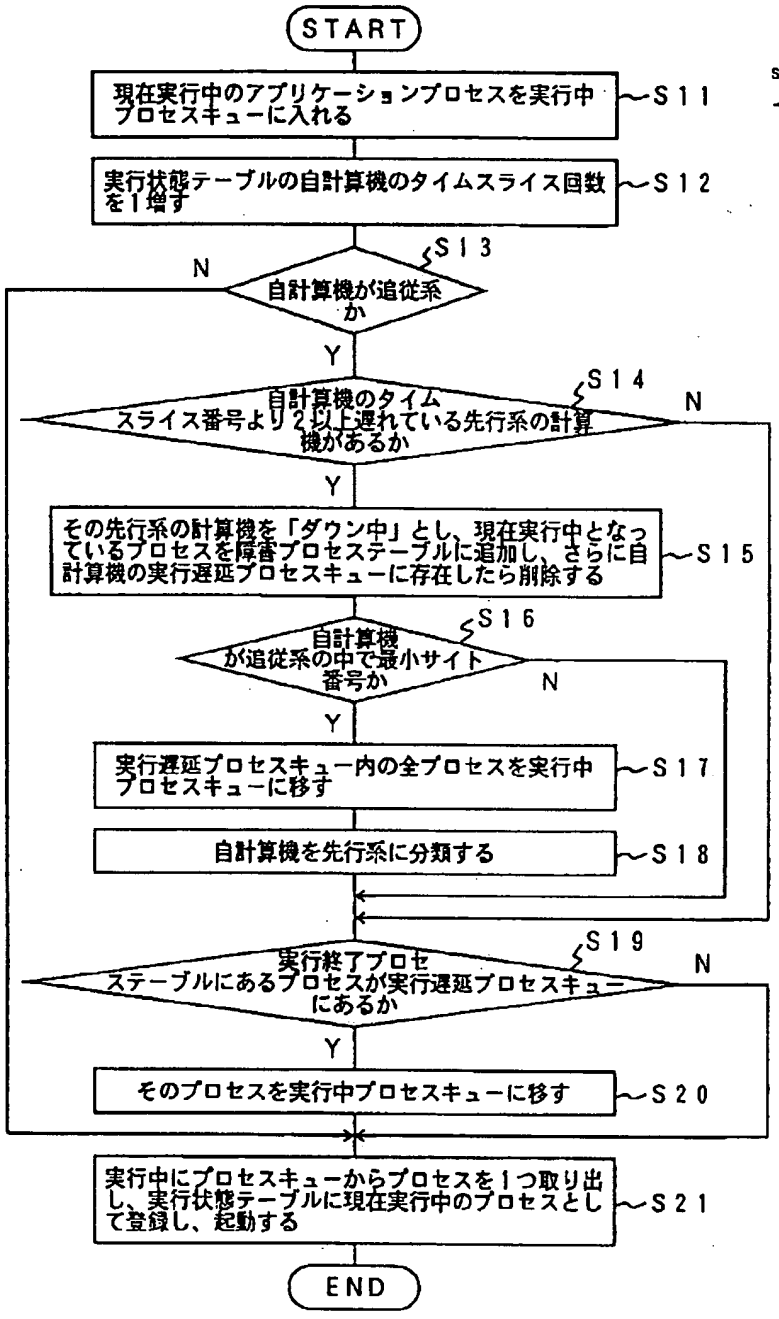


回数6のタイムスライス開始時処理後の計算機3の実行中プロセスキュー

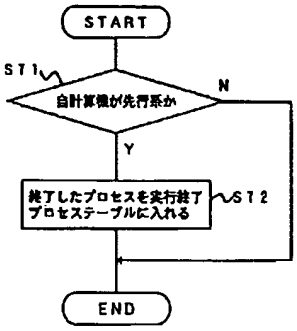
(f)

回数6のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

【図3】


START
現在実行中のアプリケーションプロセスを実行中プロセスキューに入れる
S11
実行状態テーブルの自計算機のタイムスライス回数を1増す
S12
S13
自計算機が追従系か
N
Y
S14
自計算機のタイムスライス番号より2以上遅れている先行系の計算機があるか
N
Y
その先行系の計算機を「ダウン中」とし、現在実行中となっているプロセスを障害プロセステーブルに追加し、さらに自計算機の実行遅延プロセスキューに存在したら削除する
S15
S16
自計算機が追従系の中で最小サイト番号か
N
Y
実行遅延プロセスキュー内の全プロセスを実行中プロセスキューに移す
S17
自計算機を先行系に分類する
S18
S19
実行終了プロセステーブルにあるプロセスが実行遅延プロセスキューにあるか
N
Y
そのプロセスを実行中プロセスキューに移す
S20
実行中にプロセスキューからプロセスを1つ取り出し、実行状態テーブルに現在実行中のプロセスとして登録し、起動する
S21
END

【図19】


START
S71
自計算機が先行系か
N
Y
終了したプロセスを実行終了プロセステーブルに入れる
S72
END

【図6】

(a)

計算機サイト番号	1	2	3
先行系／追従系	先行系	追従系	追従系
タイムスライス回数	1	1	1
現在実行中プロセス	P1	－	－

回数1のタイムスライス開始時処理後の実行状態テーブル

(b)

回数1のタイムスライス開始時処理後の計算機1の実行中プロセスキュー

(c)

P1

回数1のタイムスライス開始時処理後の計算機2，3の実行遅延プロセスキュー

(d)

P2

プロセスP2発生後の計算機1の実行中プロセスキュー

【図21】

(a)

計算機サイト番号	1	2	3
先行系／追従系	先行系	追従系	追従系
タイムスライス回数	0	0	0
現在実行中プロセス	－	－	－

実行状態テーブルの初期状態

(b)

P1 (高)

プロセスP1投入後の計算機1の実行中プロセスキュー

(c)

P1

プロセスP1投入後の計算機2，3の実行遅延プロセスキュー

【図7】

(a)

計算機サイト番号	1	2	3
先行系／追従系	先行系	追従系	追従系
タイムスライス回数	2	2	2
現在実行中プロセス	P2	－	－

回数2のタイムスライス開始時処理後の実行状態テーブル

(b)

P1

回数2のタイムスライス開始時処理後の計算機1の実行中プロセスキュー

(c)

P1

回数2のタイムスライス開始時処理後の計算機2，3の実行遅延プロセスキュー

(d)

P1	P3	P4

プロセスP3，4発生後の計算機1の実行中プロセスキュー

(e)

P2

プロセスP2終了後の実行終了プロセステーブル

【図13】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン中	先行系	追従系
タイムスライス回数	4	8	8
現在実行中プロセス	P3	－	P4

回数8のタイムスライス開始時処理後の実行状態テーブル

(b)

回数8のタイムスライス開始時処理後の計算機3の実行中プロセスキュー

(c)

回数8のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

【図8】

(a)

計算機サイト番号	1	2	3
先行系／追従系	先行系	追従系	追従系
タイムスライス回数	3	3	3
現在実行中プロセス	P1	－	－

回数3のタイムスライス開始時処理後の実行状態テーブル

(b)

P3	P4					

回数3のタイムスライス開始時処理後の計算機1の実行中プロセスキュー

(c)

P1						

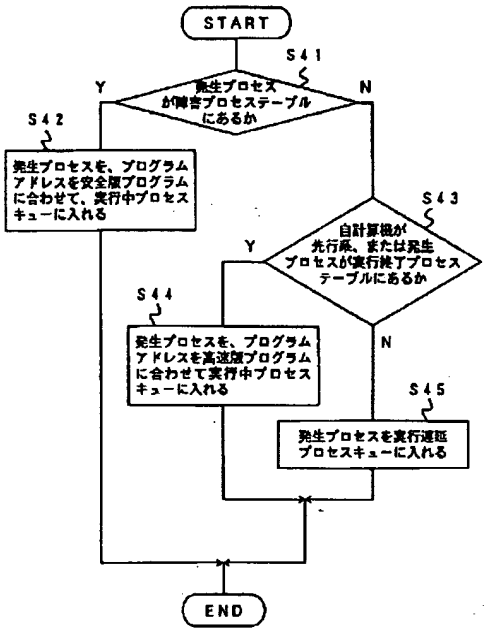
回数3のタイムスライス開始時処理後の計算機2，3の実行遅延プロセスキュー

(d)

P2，P1

プロセスP1終了後の実行終了プロセステーブル

【図17】



【図9】

(a)

計算機サイト番号	1	2	3
先行系／追従系	先行系	追従系	追従系
タイムスライス回数	4	4	4
現在実行中プロセス	P3	P1	P1

回数4のタイムスライス開始時処理後の実行状態テーブル

(b)

P4						

回数4のタイムスライス開始時処理後の計算機1の実行中プロセスキュー

(c)

回数4のタイムスライス開始時処理後の計算機2，3の実行中プロセスキュー

(d)

回数4のタイムスライス開始時処理後の計算機2，3の実行遅延プロセスキュー

(e)

P2						

プロセスP2生成後の計算機2，3の実行中プロセスキュー

【図22】

(a)

計算機サイト番号	1	2	3
先行系／追従系	先行系	追従系	追従系
タイムスライス回数	1	1	1
現在実行中プロセス	P1	－	－

回数1のタイムスライス開始時処理後の実行状態テーブル

(b)

回数1のタイムスライス開始時処理後の計算機1の実行中プロセスキュー

(c)

P1						

回数1のタイムスライス開始時処理後の計算機2，3の実行遅延プロセスキュー

(d)

P2 (高)						

プロセスP2発生後の計算機1の実行中プロセスキュー

【図10】

(a)

計算機サイト番号	1	2	3
先行系／追従系	先行系	追従系	追従系
タイムスライス回数	4	5	5
現在実行中プロセス	P3	P2	P2

回数5のタイムスライス開始時処理後の実行状態テーブル

(b)

P1						

回数5のタイムスライス開始時処理後の計算機2，3の実行中プロセスキュー

(c)

P3	P4					

プロセスP3，P4生成後の計算機2，3の実行遅延プロセスキュー

【図12】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン中	先行系	追従系
タイムスライス回数	4	7	7
現在実行中プロセス	P3	P4	－

回数7のタイムスライス開始時処理後の実行状態テーブル

(b)

回数7のタイムスライス開始時処理後の計算機2，3の実行中プロセスキュー

(c)

P4						

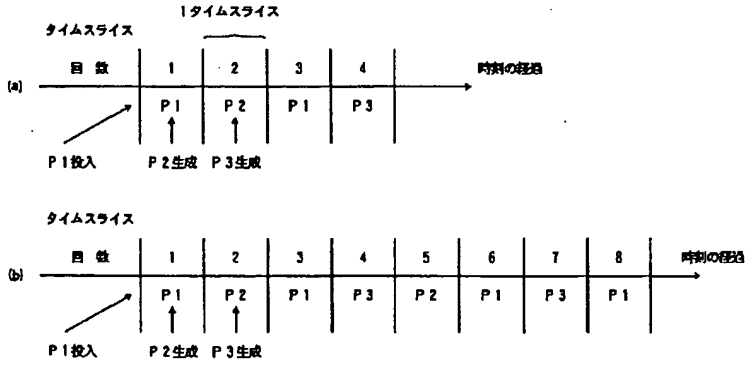
回数7のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

(d)

P2，P1，P4

プロセスP4終了後の実行終了プロセステーブル

【図20】



【図24】

(a)

計算機サイト番号	1	2	3
先行系／追従系	先行系	追従系	追従系
タイムスライス回数	2	3	3
現在実行中プロセス	P2	－	－

回数3のタイムスライス開始時処理後の実行状態テーブル

(b)

回数3のタイムスライス開始時処理後の計算機1の実行中プロセスキュー

(c)

P1						

回数3のタイムスライス開始時処理後の計算機2，3の実行遅延プロセスキュー

【図30】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン	先行系	追従系
タイムスライス回数	2	9	9
現在実行中プロセス	P2	－	P1

回数9のタイムスライス開始時処理後の実行状態テーブル

(b)

P3 (高)	P2 (安)					

回数9のタイムスライス開始時処理後の計算機3の実行中プロセスキュー

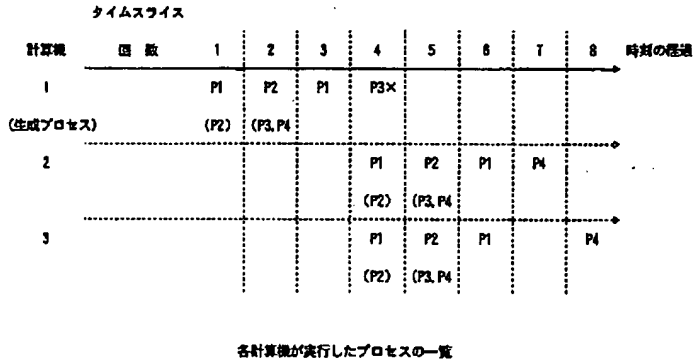
(c)

回数9のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

【図39】

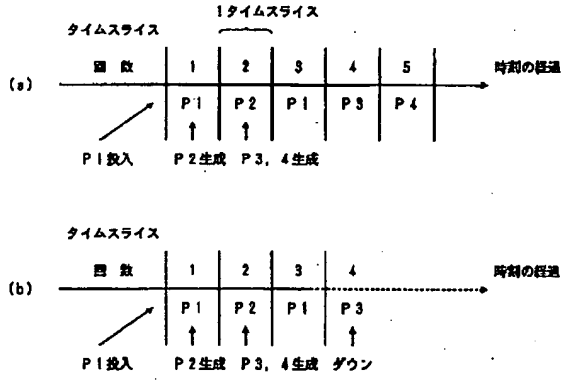
プログラム名	プログラムの開始アドレス
P001	0000
P002	1000
P003	0800
P004	1200

【図14】

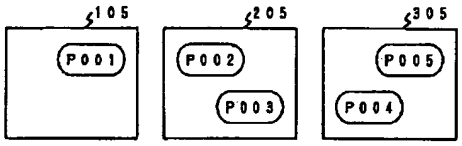


各計算機が実行したプロセスの一覧

【図15】



【図37】



【図23】

(a)

計算機サイト番号	1	2	3
先行系／追従系	先行系	追従系	追従系
タイムスライス回数	2	2	2
現在実行中プロセス	P2	－	－

回数2のタイムスライス開始時処理後の実行状態テーブル

(b)

P1 (高)						

回数2のタイムスライス開始時処理後の計算機1の実行中プロセスキュー

(c)

P1						

回数2のタイムスライス開始時処理後の計算機2，3の実行遅延プロセスキュー

【図31】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン	先行系	追従系
タイムスライス回数	2	10	10
現在実行中プロセス	P2	－	P3

回数10のタイムスライス開始時処理後の実行状態テーブル

(b)

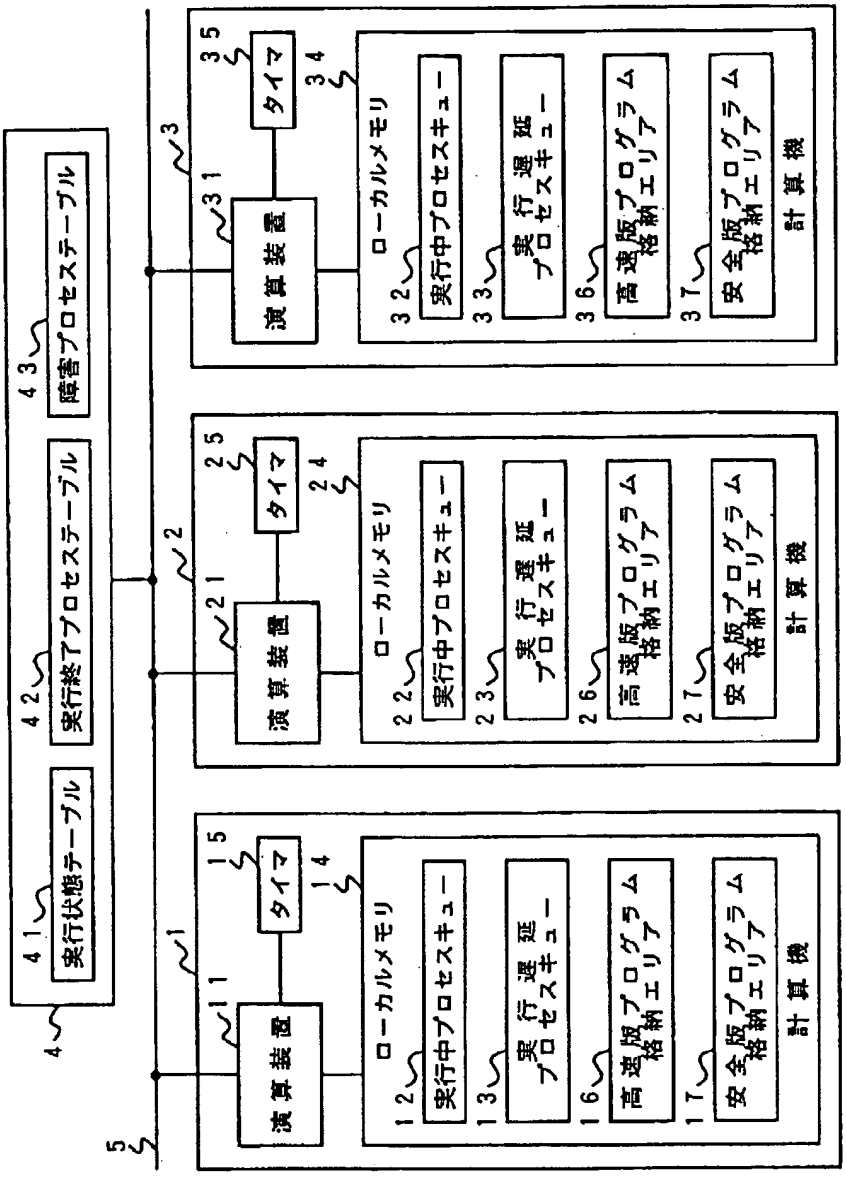
P2 (変)						

回数10のタイムスライス開始時処理後の計算機3の実行中プロセスキュー

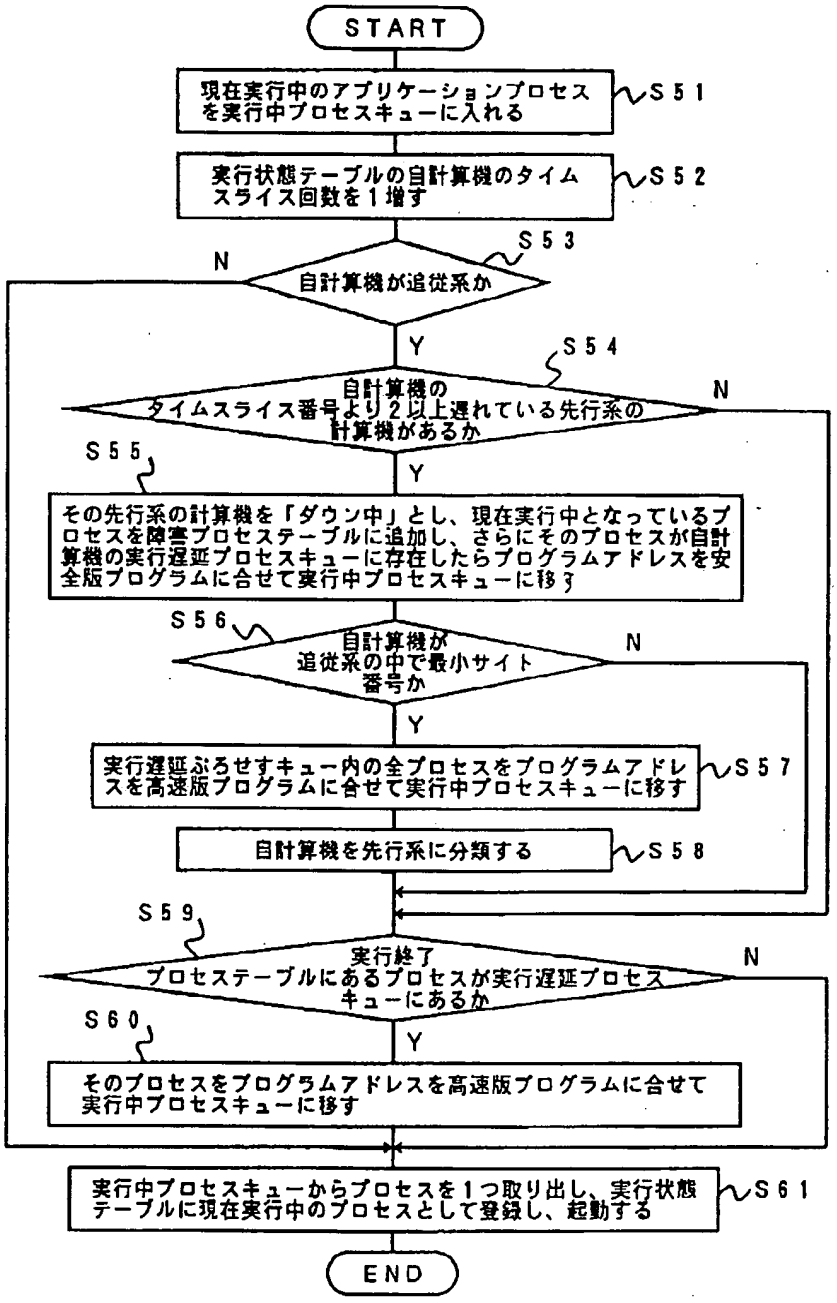
(c)

回数10のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

【図16】


4
41
実行状態テーブル
42
実行終了プロセステーブル
43
障害プロセステーブル
5
1
11
演算装置
15
タイマ
14
ローカルメモリ
12
実行中プロセスキュー
13
実行遅延プロセスキュー
16
高速版プログラム格納エリア
17
安全版プログラム格納エリア
計算機
2
21
演算装置
25
タイマ
24
ローカルメモリ
22
実行中プロセスキュー
23
実行遅延プロセスキュー
26
高速版プログラム格納エリア
27
安全版プログラム格納エリア
計算機
3
31
演算装置
35
タイマ
34
ローカルメモリ
32
実行中プロセスキュー
33
実行遅延プロセスキュー
36
高速版プログラム格納エリア
37
安全版プログラム格納エリア
計算機

【図18】


START
現在実行中のアプリケーションプロセスを実行中プロセスキューに入れる
S51
実行状態テーブルの自計算機のタイムスライス回数を1増す
S52
S53
自計算機が追従系か
N
Y
S54
自計算機のタイムスライス番号より2以上遅れている先行系の計算機があるか
N
Y
S55
その先行系の計算機を「ダウン中」とし、現在実行中となっているプロセスを障害プロセステーブルに追加し、さらにそのプロセスが自計算機の実行遅延プロセスキューに存在したらプログラムアドレスを安全版プログラムに合せて実行中プロセスキューに移す
S56
自計算機が追従系の中で最小サイト番号か
N
Y
実行遅延ふろせすキュー内の全プロセスをプログラムアドレスを高速版プログラムに合せて実行中プロセスキューに移す
S57
自計算機を先行系に分類する
S58
S59
実行終了プロセステーブルにあるプロセスが実行遅延プロセスキューにあるか
N
Y
S60
そのプロセスをプログラムアドレスを高速版プログラムに合せて実行中プロセスキューに移す
実行中プロセスキューからプロセスを1つ取り出し、実行状態テーブルに現在実行中のプロセスとして登録し、起動する
S61
END

【図25】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン	追従系	追従系
タイムスライス回数	2	4	4
現在実行中プロセス	P3	P1	－

回数4のタイムスライス開始時処理後の実行状態テーブル

(b)

回数4のタイムスライス開始時処理後の計算機2，3の実行中プロセスキュー

(c)

回数4のタイムスライス開始時処理後の計算機2の実行中プロセスキュー

(d)

P1						

回数4のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

(e)

P2

回数4のタイムスライス開始時処理後の障害プロセステーブル

(f)

P2 (変)						

プロセスP2生成後の計算機2の実行中プロセスキュー

【図26】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン	先行系	追従系
タイムスライス回数	2	5	5
現在実行中プロセス	P2	P2	－

回数5のタイムスライス開始時処理後の実行状態テーブル

(b)

P1 (高)						

回数5のタイムスライス開始時処理後の計算機2の実行中プロセスキュー

(c)

回数5のタイムスライス開始時処理後の計算機3の実行中プロセスキュー

(d)

P1						

回数5のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

(e)

P2

回数5のタイムスライス開始時処理後の障害プロセステーブル

(f)

P1 (高)	P3 (高)					

計算機2でプロセスP3生成後の計算機2の実行中プロセスキュー

【図32】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン	先行系	追従系
タイムスライス回数	2	11	11
現在実行中プロセス	P2	－	P2

回数11のタイムスライス開始時処理後の実行状態テーブル

(b)

回数11のタイムスライス開始時処理後の計算機3の実行中プロセスキュー

(c)

回数11のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

【図27】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン	先行系	追従系
タイムスライス回数	2	6	6
現在実行中プロセス	P2	P1	－

回数6のタイムスライス開始時処理後の実行状態テーブル

(b)

P3 (高)	P2 (安)					

回数6のタイムスライス開始時処理後の計算機2の実行中プロセスキュー

(c)

回数6のタイムスライス開始時処理後の計算機3の実行中プロセスキュー

(d)

P1						

回数6のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

(e)

P1

計算機2でプロセスP1終了後の実行終了プロセステーブル

【図28】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン	先行系	追従系
タイムスライス回数	2	7	7
現在実行中プロセス	P2	P3	P1

回数7のタイムスライス開始時処理後の実行状態テーブル

(b)

P2 (安)						

回数7のタイムスライス開始時処理後の計算機2の実行中プロセスキュー

(c)

回数7のタイムスライス開始時処理後の計算機3の実行中プロセスキュー

(d)

回数7のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

(e)

P1, P3

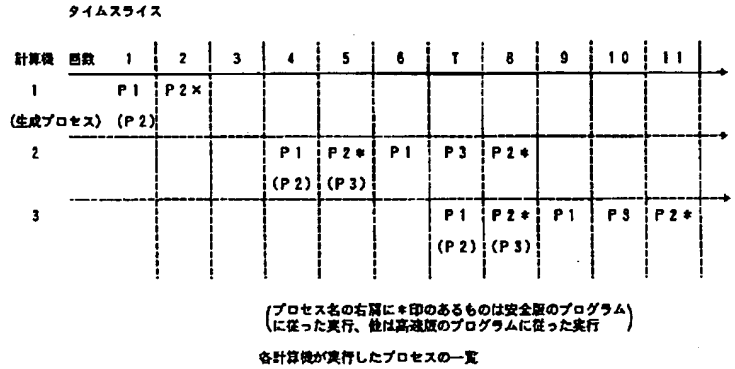
計算機2でプロセスP3終了後の実行終了プロセステーブル

(f)

P2 (安)						

計算機3でプロセスP2生成後の実行中プロセスキュー

【図33】



(プロセス名の右肩に＊印のあるものは安全版のプログラムに従った実行、他は高速版のプログラムに従った実行)

各計算機が実行したプロセスの一覧

【図29】

(a)

計算機サイト番号	1	2	3
先行系／追従系	ダウン	先行系	追従系
タイムスライス回数	2	8	8
現在実行中プロセス	P2	P2	P2

回数8のタイムスライス開始時処理後の実行状態テーブル

(b)

回数8のタイムスライス開始時処理後の計算機2の実行中プロセスキュー

(c)

P1 (高)						

回数8のタイムスライス開始時処理後の計算機3の実行中プロセスキュー

(d)

回数8のタイムスライス開始時処理後の計算機3の実行遅延プロセスキュー

(e)

P1, P3, P2

計算機2でプロセスP2終了後の実行終了プロセステーブル

(f)

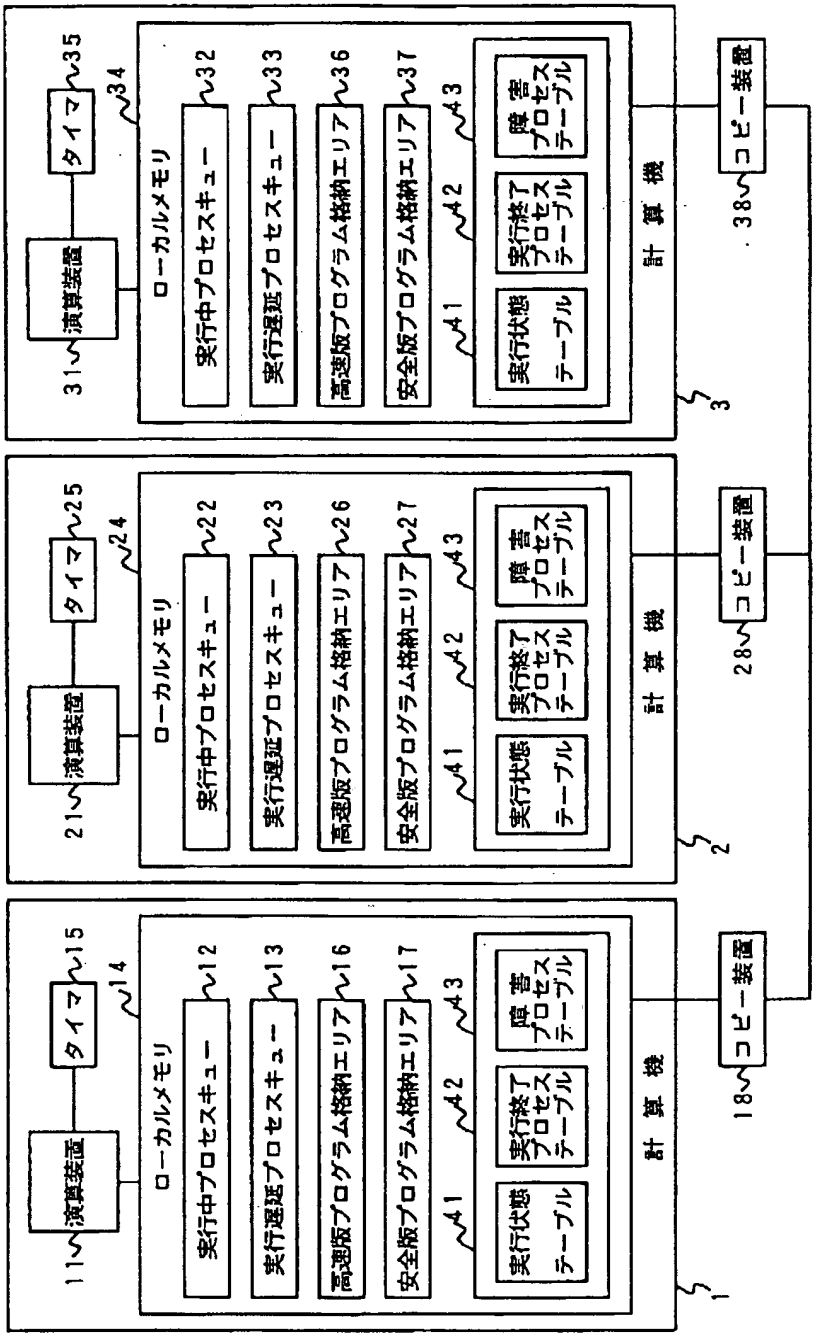
P1 (高)	P3 (高)					

計算機3でプロセスP3生成後の実行中プロセスキュー

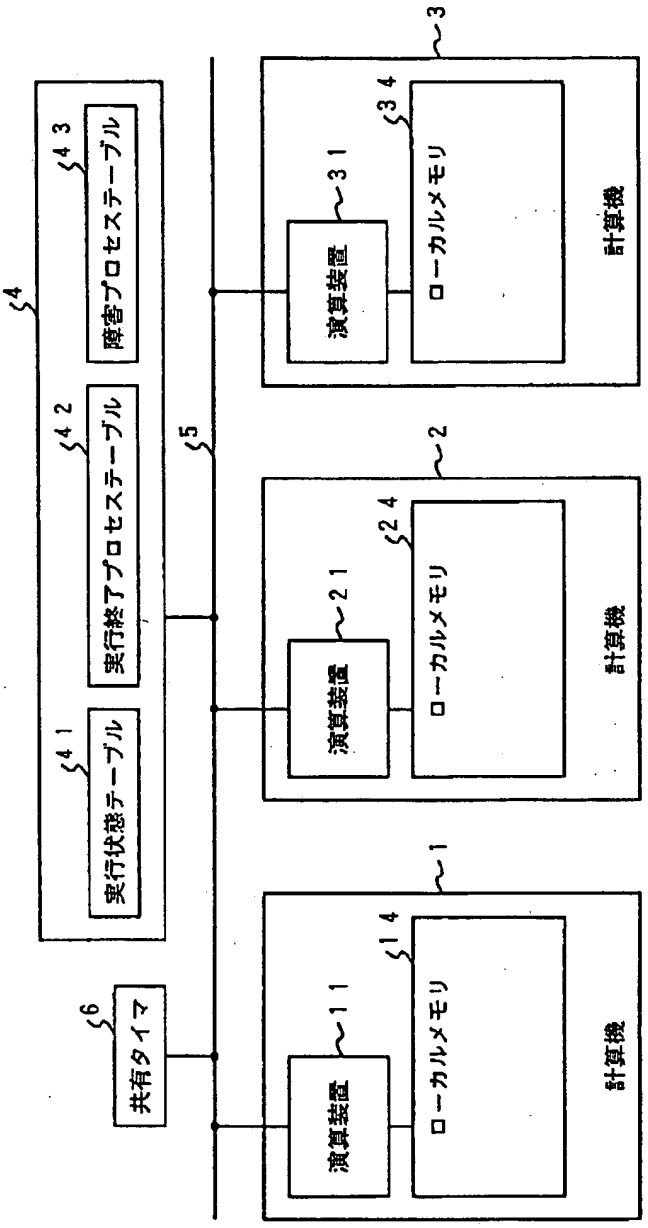
【図38】

プログラム名	処　理　機　能
P001	2つのベクトルの和を求め、その絶対値の2乗値を計算する
P002	2つのベクトルの和を計算する
P003	ベクトルの大きさを計算する
P004	スカラーの2乗を計算する
P005	スカラーの積算値を計算する

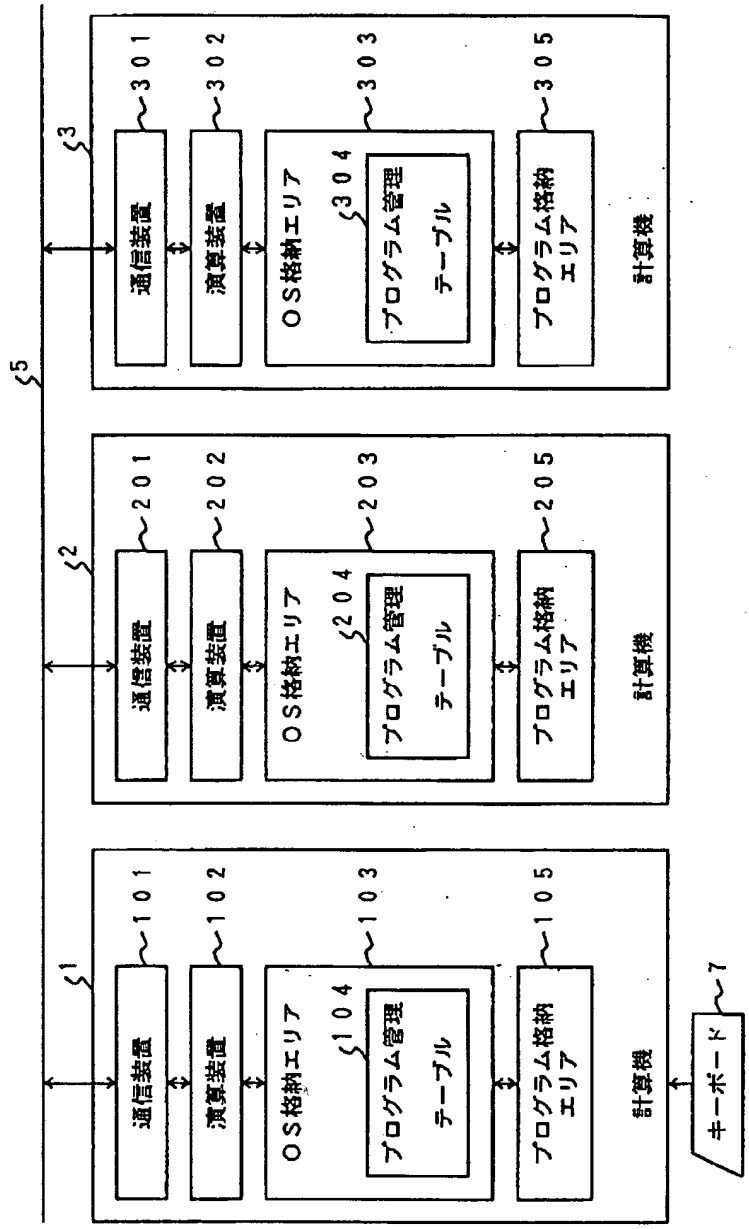
【図34】


計算機
1
2
3
演算装置
11
21
31
タイマ
15
25
35
ローカルメモリ
14
24
34
実行中プロセスキュー
12
22
32
実行遅延プロセスキュー
13
23
33
高速版プログラム格納エリア
16
26
36
安全版プログラム格納エリア
17
27
37
43
実行状態テーブル
41
実行終了プロセステーブル
42
障害プロセステーブル
コピー装置
18
28
38

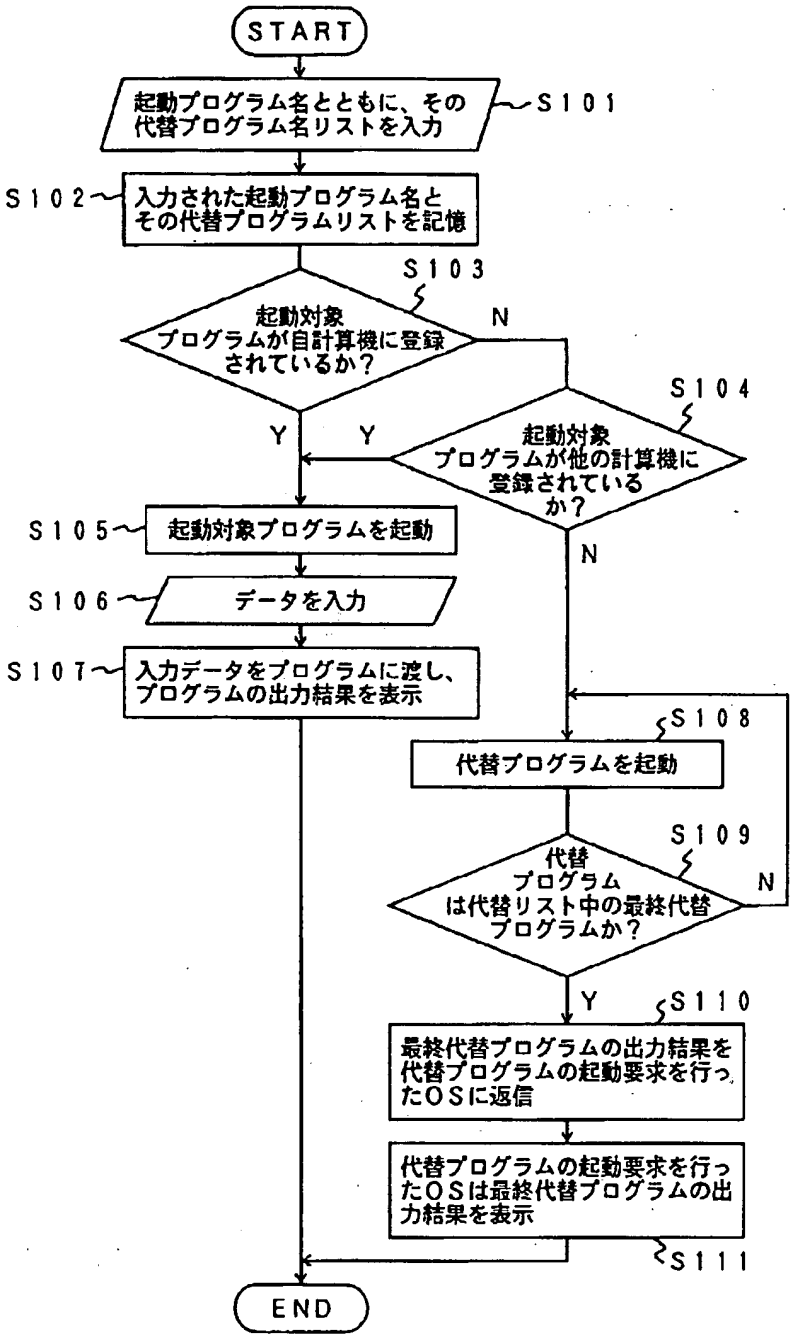
【図35】


4
41
実行状態テーブル
42
実行終了プロセステーブル
43
障害プロセステーブル
5
6
共有タイマ
1
11
演算装置
14
ローカルメモリ
計算機
2
21
演算装置
24
ローカルメモリ
計算機
3
31
演算装置
34
ローカルメモリ
計算機

【図36】


1
2
3
5
7
通信装置
101
演算装置
102
OS格納エリア
103
104
プログラム管理テーブル
プログラム格納エリア
105
計算機
キーボード
201
202
203
204
205
301
302
303
304

【図40】


START
起動プログラム名とともに、その代替プログラム名リストを入力
S101
S102
入力された起動プログラム名とその代替プログラムリストを記憶
S103
起動対象プログラムが自計算機に登録されているか？
N
Y
S104
起動対象プログラムが他の計算機に登録されているか？
Y
N
S105
起動対象プログラムを起動
S106
データを入力
S107
入力データをプログラムに渡し、プログラムの出力結果を表示
S108
代替プログラムを起動
S109
代替プログラムは代替リスト中の最終代替プログラムか？
N
Y
S110
最終代替プログラムの出力結果を代替プログラムの起動要求を行ったOSに返信
代替プログラムの起動要求を行ったOSは最終代替プログラムの出力結果を表示
S111
END